A Principled Approach to Managing Routing in Large ISP Networks

Yi Wang

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
By the Department of
Computer Science

Advisor: Professor Jennifer Rexford

June 2009

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
JUN 2009		**00-00-2009 to 00-00-2009**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Principled Approach to Managing Routing in Large ISP Networks	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Princeton University,Department of Computer Science,Princeton,NJ,08544**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Internet Service Providers (ISPs) are the core building blocks of the Internet, and play a crucial role in keeping the Internet well-connected and stable, as well as providing services that meet the needs of other ASes (and their users). As a result, an ISP plays different roles in its operation: (1) as part of the Internet, an ISP is expected to help keep the global network stable; (2) when interacting with neighboring networks, an ISP faces diverse requirements from different neighbors about the kinds of routes they prefer; and (3) internally, an ISP needs to maintain and upgrade its own network periodically, and wants avoid disruptions during those operations as much as possible. As the Internet has become an integral part of the world?s communications infrastructure, today?s ISPs face a number of routing management challenges at these different scopes, which include: (i) maintaining the stability of the global Internet while meeting the increasingly demands for providing diverse routes from its customers, (ii) supporting more flexible routing policy configuration in bilateral contractual relationships with its neighbors, and (iii) making network maintenance and other network management operations in their own networks easier and less disruptive to routing protocols and data traffic. This dissertation takes a principled approach to addressing these challenges. We propose three abstractions that guide the design and implementation of our system solutions. First, we propose the abstraction of a ?neighbor-specific route selection problem? and a corresponding ?Neighbor- Specific BGP? (NS-BGP) model that capture the requirement of customized route selection for different neighbors. Since one ISP?s route selection decisions could cause the global Internet to become unstable, we prove the conditions under which the Internet is guaranteed to remain stable even if individual ISPs make the transition to this more flexible route-selection model. Second, we model policy configuration as a decision problem, which offers an abstraction that supports the reconciliation of multiple objectives. Guided by this abstraction and the Analytic Hierarchy Process, a decision-theoretic technique for balancing conflicting objectives, we designed and implemented a prototype of an extensible routing control platform (Morpheus) that enables an ISP to select routes for different neighbors individually and make flexible trade-offs among policy objectives through a simple and intuitive configuration interface. Finally, we propose the abstraction of the separation between ?physical? and ?logical? configurations of routers, which leads us to the design and prototype implementation of ?virtual router migration? (VROOM), a new, generic technique to simplify and enable a broad range of network management tasks, from planned maintenance to reducing power

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**147**	

© Copyright by Yi Wang, 2009. All rights reserved.

Abstract

Internet Service Providers (ISPs) are the core building blocks of the Internet, and play a crucial role in keeping the Internet well-connected and stable, as well as providing services that meet the needs of other ASes (and their users). As a result, an ISP plays different roles in its operation: (1) as part of the Internet, an ISP is expected to help keep the global network stable; (2) when interacting with neighboring networks, an ISP faces diverse requirements from different neighbors about the kinds of routes they prefer; and (3) internally, an ISP needs to maintain and upgrade its own network periodically, and wants avoid disruptions during those operations as much as possible. As the Internet has become an integral part of the world's communications infrastructure, today's ISPs face a number of routing management challenges at these different scopes, which include: (i) maintaining the stability of the *global* Internet while meeting the increasingly demands for providing diverse routes from its customers, (ii) supporting more flexible routing policy configuration in bilateral contractual relationships with its *neighbors*, and (iii) making network maintenance and other network management operations in their *own* networks easier and less disruptive to routing protocols and data traffic.

This dissertation takes a principled approach to addressing these challenges. We propose three abstractions that guide the design and implementation of our system solutions. First, we propose the abstraction of a "neighbor-specific route selection problem" and a corresponding "Neighbor-Specific BGP" (NS-BGP) model that capture the requirement of customized route selection for different neighbors. Since one ISP's route selection decisions could cause the global Internet to become unstable, we prove the conditions under which the Internet is guaranteed to remain stable even if individual ISPs make the transition to this more flexible route-selection model. Second, we model policy configuration as a decision problem, which offers an abstraction that supports the reconciliation of multiple objectives. Guided by this abstraction and the Analytic Hierarchy Process, a decision-theoretic technique for balancing conflicting objectives, we designed and implemented a prototype of an extensible routing control platform (Morpheus) that enables an ISP to select routes

for different neighbors individually and make flexible trade-offs among policy objectives through a simple and intuitive configuration interface. Finally, we propose the abstraction of the separation between "physical" and "logical" configurations of routers, which leads us to the design and prototype implementation of "virtual router migration" (VROOM), a new, generic technique to simplify and enable a broad range of network management tasks, from planned maintenance to reducing power consumption. Collectively, the contributions of the dissertation provide simple system solutions for an ISP to autonomously manage its routing more flexibly and effectively without affecting global routing stability.

Acknowledgments

I am extremely fortunate to have Jennifer Rexford as my advisor, working with her has made my four years at Princeton one of the best times of my life. I cannot remember how many times I told myself, my family and friends that my advisor is the best imaginable and I could not believe how lucky I am. I want to thank Jennifer for being both “hands off” and “hands on”: giving me enough freedom and encouraging me to pursue problems I found most interesting, yet providing inspirational guidance and detailed help whenever I need them, e.g., bouncing off new ideas, deep discussion of research approaches, learning how to improve paper writing and give better presentation, etc. Her sharp intellect and solid expertise made the discussions I had with her some of the most enlightening and memorable moments of my graduate career, without which this dissertation would not have been possible. I am also very grateful for Jennifer’s encouragement and support for me to present my work at various conferences, universities and research institutions, and to obtain broader experience through internships. Thanks to these wonderful experiences, I built confidence from presenting to and discussing with many brightest minds and experts in the field, got invaluable feedback for my work, deepened understanding of the academic and industrial research landscape, and made many good friends. Besides being a fantastic advisor, Jennifer continually amazed me with her thoughtfulness and genuine kindness to everyone she met, and her humility. She will be role model for me for years to come.

I am also very lucky to have worked extensively with Kobus van der Merwe. As mentor of my internship at AT&T Research, Kobus shared my passion for practical problems. His openness to new ideas and humor made him a fun person to work with. As paper co-author, his provocative thinking and rigorous attitude set an example of a true scholar that I strived to follow. As reader of my dissertation, he read this dissertation with vigilance, and his detailed suggestions helped me strengthen many aspects of this dissertation.

I would like to thank Mike Freedman, Larry Peterson and Ed Felten for serving on my thesis committee, and giving me valuable feedback on my work.

I owe a special thank you to Ioannis Avramopoulos, Eric Keller, Brian Biskeborn and Michael Schapira for their invaluable help and contribution to the papers we co-authored, which constitute the major chapters of this dissertation. Collaborating with them has been a truly pleasant and inspirational experience. I would also like to thank Nick Feamster, Matthew Caeser, Aman Shaikh, Tim Griffin, David Walker, Murtaza Motiwala, Vytautas Valancius, Dan Wendlandt, Ricardo Oliveira, Mohit Lad and many others for their valuable feedback and help to my research over the years.

My research was supported by the National Science Foundation, Homeland Security Advanced Research Projects Agency/ARO and Princeton. Many thanks to all for making this research possible.

My life at Princeton would have been unimaginable without an amazing group of friends. As part of the Cabernet group, I benefited tremendously from its members' diverse interests and expertise, in research and beyond. Changhoon Kim was my guru of enterprise networking and gave me great advice and support for my research and beyond in countless discussions. He also introduced me to mountain biking, together with Mark McCann. Haakon Ringberg was a reliable source of great comments and suggestions for a lot of things: my research papers and presentations, wines, good restaurants, just to name a few. I had a great time living, running, cooking, hiking, and BBQ-ing together with Chang and Haakon in the summer of 2007, when we were interning at AT&T Research. I will miss Sharon Goldberg's right-on criticism of my presentations followed by constructive comments. I would also like to thank Jiayue He, Rui Zhang-Shen, Eric Keller, Minlan Yu, Yaping Zhu, Elliott Karpilovsky, Martin Suchara, Wenjie Jiang, Pete Hummon, Alex Fabrikant for fun conversations and their invaluable feedback for my work over the years.

I would also like to thank my friends from the rest of the Computer Science Department (Zafer Barutcuoglu, Chris Bienia, Jordan Boyd-Garber, Joe Calandrino, Tony Capra, Melissa Carroll, Will Clarkson, Forrester Cole, Wei Dong, Ariel Feldman, Rebecca Fiebrink, Shirley Gaw, Michael Golightly, William Josephson, Wyatt Lloyd, Xiaojuan Ma, KyoungSoo Park, Lindsey Poole, Sid Sen, David Shue, Umar Syed, Jeff Terrace, Mike Wawrzoniak, Qian Xi, Harlan Yu, Bill Zeller,

Yun Zhang) for introducing me to new research areas and making me feel home in the department. I thank my friends from the Princeton Graduate Student Government and the GSG Events Board (Kostas Aisopos, Raja Chahal, Jeff Dwoskin, Christina Holtholm, Manos Koukoumidis, Jenna Losh, Olivia Matel, Marina Paul, Lisa Schreyer, Ashley Thrall, Anne Twitty), the Princeton Graduate Engineering Council (Theresa Cho, Brian Figura, Brian Ellis, Raghuveer Vinukollu, Vijay Krishnamurthy), the Princeton Badminton Club (Shelley Chan, Kevin Fan, Teck Hsien Ho, Chern Han Lim, Richard She, Lova Sun, Shu Haur Tang, Aldi Wahyudi), and too many others to mention, for making my life outside research fun and enjoyable.

I am grateful to my host family Dotty Westgate for introducing me to American culture through Christmas and Super-bowl parties and delightful conversations. I thank Jingrong Huang and Da Teng for helping me settle down when I first arrived at Princeton. I am also indebted to Natasha Haase for her invaluable help to my job search. Special thanks to Melissa Lawson for her help throughout my study at Princeton, specially during the graduation process.

I thank my parents, Zhiying Sun and Baimao Wang, for their enduring love, belief and encouragement, without which this accomplishment would have been impossible.

Above all, I would like to thank my wife, Jing Cao, for her love, support, and unwavering faith in me, and for bringing me joy and optimism throughout my Ph.D. I dedicate this dissertation to her.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Since its commercialization in the early 1990s, the Internet has experienced exponential growth and phenomenal success. Evolving from the then government-owned and -operated NSFNet accessible only by academic researchers in the U.S., today's Internet consists of tens of thousands of independently operated networks that offer communication services to billions of people around the globe.

Many of these networks in the Internet are Internet Service Providers (ISPs), which offer other networks (i.e., their customers) access to the Internet. Collectively acting as the "core" of the Internet, ISPs play a crucial role in keeping the Internet well-connected and stable, as well as providing network services that meet the needs of other networks (and their users). All the services offered by ISPs fundamentally rely on *routing*, the process of discovering paths in a network along which to send traffic to reach other destinations.

Managing routing is essential in ISPs' network operation. By configuring the many routers in its network, an ISP implements policies that reflect its business relationships with neighboring networks, and adjusts routing protocols to select paths with desirable properties. If an ISP is able to provide the paths that meet its customers' needs (e.g., low latency / stable / secure) and manage its network to provide reliable service, it is likely to become commercially successfully by retaining

existing customers and attracting new ones. However, if an ISP fails to properly manage its routing, it will eventually lose its customers.

In spite of its obvious importance, today's ISP routing management practices are surprisingly primitive. For example, even though different networks today may have very different preferences for the kinds of paths they would like to use (e.g., a financial institution may prefer the most secure paths that do not traverse any untrusted networks, whereas a provider of online gaming or voice-over-IP service may prefer paths with the lowest latency), today's ISPs simply are not capable of providing such customized routing services—a router is only allowed to select a *single* best path and only that path may be offered to its neighbors. Even with the routing operations that can be done today, the configuration practices are usually dauntingly complicated, error-prone, and disruptive. For example, even tasks as routine as planned maintenance of routers causes disruption to routing protocols and user traffic. Despite the fact that ISPs typically spend 3-5 times more money on network operation than on equipment [75, 74], and about 70% of the money spent on operation is spent on network management [60], most of network outages are caused by operators errors rather than equipment failure [60].

Given that the current routing management practices are both inadequate and overly-complex, addressing both problems at the same time is a challenging task. Rather than proposing new point solutions to these problems, this dissertation takes a principled approach: we first identify the root causes of the various problems in today's ISP routing management, and then propose a set of principles to treat these root causes. Using these principles as a guide, we design and implement system solutions that offer ISPs more flexibility in realizing routing policies and, at the same time, are simple, intuitive to configure, and minimize disruption. The system solutions we propose can be autonomously deployed by individual ISPs without changing the format of any routing protocol messages or requiring collaboration with neighboring ASes.

We first give a high-level overview of ISP routing management to introduce the goals ISPs try to achieve by managing routing in their networks. We then discuss three big challenges ISPs

face today in meeting these goals in Section 1.2, and summarize the major contributions of this dissertation in Section 1.3.

1.1 An Overview of ISP Routing Management

The Internet is a network of tens of thousands of independently owned and operated networks known as *Autonomous Systems* (ASes). To achieve and maintain global connectivity, these ASes share reachability information by using a *routing protocol* as the "common language"—the Border Gateway Protocol (BGP) [80]. BGP is a path-vector protocol. Every BGP route contains a complete AS-level routing path (i.e., AS path) to reach a set of destinations (e.g., a block of IP addresses known as "prefixes"). BGP is also a single-path protocol. Even if a router learns multiple paths to reach a particular destination (as is often the case for edge routers of large ISPs), it must select a *single* best route and may only announce this best route to neighboring routers. Two adjacent ASes exchange their best routes through through their edge routers. Besides BGP, each AS also runs an internal routing protocol ("Interior Gateway Protocol", or IGP) to disseminate reachability information about destinations within its network. Together, IGP and BGP ensure an internal router of an AS knows how to send traffic first to an edge router of the network (via an IGP path), in order to reach its final destination many AS-hops away (via a BGP path).

Being an AS that provides Internet access to other neighboring ASes, an Internet Service Provider (ISP) plays three different roles in its operation. The different requirements associated with these roles add to the complexity of the routing management problem an ISP deals with every day.

1.1.1 An ISP's Role in The Global Internet

First of all, as a participant of the global Internet, an ISP has the obligation to keep the Internet stable. Given the way BGP works, the routes chosen by one AS are directly affected by the routes

chosen and announced by its neighbors, which, in turn, are affected by their neighbors' decisions. As a result of this "chain effect" of route selection and propagation, any local routing changes introduced by one ISP can have a global impact on the stability of the Internet. Unfortunately, the BGP protocol itself does not guarantee routing stability [63, 64, 49], and there is tension between the flexibility of local path selection and global stability properties [49, 43, 42]. Certain local configuration changes may lead to permanent global route oscillation—an anomalous situation in which a set of ASes announce, withdraw and then re-announce some routes indefinitely—that can cause significant disruption and performance degradation to data traffic in many networks (including the ISP that causes the oscillation) [63, 64]. Therefore, an ISP has the responsibility and incentive to make sure that any changes an ISP introduces to its local routing policy configuration do not undermine the stability of the global Internet.

1.1.2 An ISP's Interaction With Its Neighbors

Each ISP is an independent entity with its own economic interest. Since the commercialization of the Internet in the early 1990s, ISPs have formed an economic ecosystem that is built on different bilateral business relationships between a pair of neighboring ASes, among which the two most common ones are "customer-provider" relationships and "peer-to-peer" relationships. In the "customer-provider" relationship, a customer AS pays its provider AS for connectivity to the rest of the Internet, whereas in the "peer-to-peer" relationship, peer ASes carry traffic between their respective customers free of charge (if the two peers exchange roughly equal amounts of traffic).

These financial arrangements have a direct influence on how ASes select and export routes. For example, an AS has the incentive to prefer *customer routes* (i.e., routes learned from a customer) over peer or provider routes in route selection, as the former brings it revenue. Due to similar financial reasons, an AS only exports customer routes to its peers or providers (as it gets paid by the customers to do so), but does not export peer or provider routes to other peers or providers (so that it does not carry other peer or provider traffic for free).

Thanks to these business relationships and the pursuit of economic interests, the competition for customers among ISPs has had a significant contribution to the increasing availability and price reduction of Internet access. Today, as the Internet supports an increasingly wide range of applications (from real-time applications such as Voice-over-IP and online gaming to security-sensitive applications such as online shopping and online banking), an ISP faces diverse requirements from different customers about the kinds of routes they want. For example, customers in the financial industry may prefer the most secure routes (e.g., routes that do not traverse some untrusted ASes), whereas customers hosting interactive applications like online gaming or voice-over-IP may prefer paths with low latency. If such options were available, they might be willing to pay a higher price to have the routes they want. Yet there are many other customers who may be perfectly happy with whatever paths the ISP provides at a relatively low price. In today's competitive market environment, an ISP has strong incentive to meet these diverse requirements in order to be more attractive to its customers.

1.1.3 An ISP's Responsibility in Managing Its Own Network

Finally, an ISP needs to manage its network well to achieve good performance with relatively low cost. Network management involves a range of tasks including routing policy configuration, planned equipment maintenance and upgrades, new service deployment, etc. Due to their different goals, these tasks interact with routing in very different (sometimes opposite) ways. Routing policy configuration is the *active* process of tuning the routing protocols to realize the ISP's business relationships with its neighbors and other objectives (e.g., avoid traffic congestion on any network links). On the other hand, the goal of planned maintenance is to have the job done with as *little* disruption to the routing protocols and user traffic (which is directly affected by routing) as possible. In fact, the service level agreements (SLAs) an ISP signs with its customers as part of the contract specifies the maximum downtime its service is allowed to have. Given that planned maintenance is performed frequently in large ISPs (e.g., on a daily basis), an efficient, non-disruptive mechanism

is especially desirable. Moreover, all network management tasks should introduce as few human errors (e.g., mis-configured routing policies) as possible, as these errors could significantly impact the perceived performance of an ISP's immediate neighbors and beyond, and may even cause global routing instability.

1.2 Challenges in ISP Routing Management

Almost all routing management tasks in an ISP are handled through routing *configurations*. As a result, the difficulties network operators face in configuring their networks precisely indicate the problems in today's ISP routing management practices. Among these difficulties, we observe three big problems that are most important and challenging: (1) many useful routing policies *cannot* be realized (through configuration), (2) policies that can be realized are *hard* to configure, (3) many (re)configurations routinely performed by network operators are *unnecessary* and cause disruption to routing protocols and data traffic.

1.2.1 Many Useful Routing Policies Cannot Be Realized

Large ISPs today usually learn multiple interdomain routes for the same destination at different edge routers [103]. Despite this great opportunity to select different routes for different neighbors, and the strong incentives from both the ISPs and their customers to have customized routing service, such flexible routing policies simply cannot be realized today. In the current routing architecture, routers are restricted to selecting only one best route per destination, and may only propagate that route to its neighbors. As a result, even if an ISP as a whole learns multiple routes for a destination, each individual router may not see many of the available routes, significantly reducing the number of alternative routes it can choose from.

In addition to the "route visibility" problem routers have, the BGP decision process, which selects the best route for each prefix, is also highly restrictive. It imposes a strict ranking on

the attributes of BGP updates, where the "local preference" attribute has strict priority over the "AS-path length" attribute and so on. This makes policies that strike a trade-off between different policy objectives (e.g., business relationships vs. stability, or security vs. performance) impossible. Moreover, the BGP decision process selects routes only based on the attributes of the BGP updates, falling short of realizing routing policies that, for example, require using outside measurement data.

Finally, the transition from the current "one-route-fits-all" route selection model to the more appealing customized route selection model is an ambitious shift with many questions to be answered, such as "Will it require changes to the BGP protocol?", "Can individual ISPs make this transition alone without requiring cooperation from neighboring domains?" However, besides all the system design and implementation issues, an arguably more fundamental and important question should be answered: "Would such increased flexibility of individual ISPs' *local* policies cause the *global* routing system to become unstable?" Given the importance of global routing stability, a clear answer to this question is crucial to know if any improvement to policy flexibility and customized route selection would be safe and useful in practice. However, the answer is not obvious, and may even seems a bit pessimistic at first glance. This is because even without customized route selection, today's BGP can easily oscillate, depending on the local policies ASes apply in selecting and exporting routes [63, 64, 49, 47].

Over the years, researchers have developed a reasonably good understanding of the trade-offs between local flexibility and global stability [49, 47, 43, 42, 35]. Rather than relying on Internet-wide coordination, researchers searched for practical constraints on local policies that would ensure global stability. The best known BGP stability conditions are the "Gao-Rexford conditions" that specify constraints on an AS's preferences in selecting routes, its export policies, and the topology of the network it is in [43]. The "Gao-Rexford" conditions reflect common business practices in today's Internet, which may explain why the interdomain routing system is generally stable in practice. However, these conditions may be too restrictive for ISPs to offer customized route selection. In particular, an ISP may want to violate its "preference condition" to (1) have different

preferences for different neighbors and (2) prefer peer or provider routes over customer routes for some (high-paying) customers. Whether such violation would cause global routing to become unstable is an important unanswered question.

1.2.2 Many Policies Which Are Realizable Are Hard To Configure

Besides the many useful policies that cannot be realized today, the current routing architecture and configuration interface also make many routing policies that are feasible today notoriously hard to configure, for two major reasons.

First, there is a mismatch between the level at which routing policies are *specified* and the level at which they are *implemented* today. On the one hand, an ISP's routing policies state its *AS-level* objectives, i.e., how the network as a whole should behave. On the other hand, policies are often implemented by configuring individual routers, making realizing routing policies a *router-level* operation. This mismatch raises the question of how a distributed collection of routers can realize a network-wide policy. In practice, network operators are forced to retrofit AS-level policies into router-level configurations. Techniques necessary to scale routing in large ISPs (with hundreds of routers) introduce additional challenges to this process, making the job of "translating" policies into router configurations even more daunting and error-prone. In fact, this mismatch has been shown to be the root cause of many common policy violations in practice [81].

Second, the current BGP configuration interface forces an ISP's policy objectives (e.g., business relationships, traffic management) to be specified in an intertwined and unnatural way. BGP uses a step-by-step decision process that compares all alternative routes one attribute at a time, eliminates routes that are less preferred, and stops when there is only one best route left. Among all the attributes affecting the outcome of route selection, only the "local preference" attribute (at the first step of the decision process) is completely controlled by the local AS. As a result, network operators are forced to use it for many different purposes. For example, it is common practice to use "local preference" (LOCAL_PREF) to realize business relationship policies, where customer

routes are assigned with higher LOCAL_PREF value than peer routes, which, in turn, are assigned with higher LOCAL_PREF value than provider routes. At the same time, "local preference" is also used for traffic engineering and other policy objectives. The overloading of the BGP attribute is just one of many "hacks" network operators are forced to use to realize policies via the current configuration interface.

1.2.3 Many Unnecessary Routing (Re)configurations Cause Disruption

In addition to routing configurations that realize certain policies, network operators routinely (re)configure their routers only to *facilitate* planned network maintenance. To reduce the impact to routing protocols and traffic forwarding when a router needs to be taken offline for maintenance (e.g., replacing a broken power adaptor, or fixing a malfunctioning component), it is common practice to first reconfigure the IGP in the network (e.g., increase the weights of the links adjacent to the router to make them less attractive) in order to make other routers to switch to paths that does not go through this router. After the maintenance, network operators need to configure the IGP again to restore the routing back to the state before the maintenance.

In this so-called "cost-in/cost-out" technique [93] before and after planned maintenance, routing (re)configurations are *not* the goal, but merely the *tool* to reduce the impact of the maintenance operation. However, this tool, serving as a patchwork to support planned maintenance, is far from satisfactory. Because it involves routing protocol reconfiguration, both BGP and IGP have to reconverge to a new stable state. This is especially harmful in the case of BGP as it converges significantly more slowly than IGPs. During the BGP reconvergence process, which may take as long as fifteen minutes [62, 65], traffic being sent through the ISP to and from neighboring networks may be lost or experience significant performance degradation. Moreover, planned maintenance is a routine operation that happens frequently in large ISPs. Using unnecessary and disruptive routing reconfiguration as a tool for such basic tasks not only significantly increases the cost of network management (especially such operations are often conducted semi-manually), but also introduces

more opportunities for configuration errors.

This dissertation focuses on addressing these three important challenges that mainly concern how an ISP manages its routing while interacting with its neighbors. There are other network management tasks an ISP performs in isolation from other networks, such as managing how traffic is sent through its network (i.e., traffic engineering) and ensuring security (e.g., by detecting and blocking anomalous traffic), etc. Other work provides more detailed treatment on these topics [8, 38, 40, 58, 101, 66, 61, 89, 112, 18].

1.3 Contributions

In this dissertation, we take a principled approach to addressing these limitations and challenges of today's routing practices, in an effort to enable individual ISPs to realize more flexible local policies without affecting global stability and simplify routing management operations such as policy configuration and planned maintenance. We propose three abstractions that guide the design and implementation of our system solutions, and make three major contributions.

1.3.1 A Customized Route Selection Model With Improved Stability

First, we propose the abstraction of a "neighbor-specific route selection problem" and a corresponding "Neighbor-Specific BGP" (NS-BGP) model that captures the requirement of customized route selection for different neighbors. As a modest extension to BGP, NS-BGP enables a much wider range of local policies without compromising global stability. Whereas a conventional BGP-speaking router selects a single "best" route (for each destination prefix), NS-BGP allows a router to customize the route selection on behalf of each neighbor. For example, one neighbor may prefer the shortest route, another the most secure route, and yet another the least expensive route. Surprisingly, we prove that the much more flexible NS-BGP is guaranteed to be stable under much *less* restrictive conditions on how routers "rank" the candidate routes. We also show that it is safe

to deploy NS-BGP incrementally, as a routing system with a partial deployment of NS-BGP is guaranteed to be stable, even in the presence of failure and other topology changes [105].

1.3.2 A System for Flexible Routing Configuration With Intuitive Interface

Second, we propose the abstraction of "policy configuration as a decision problem of reconciling multiple objectives". Guided by this abstraction, we have designed and implemented a prototype of an extensible routing control platform (Morpheus) that supports NS-BGP, and enables a single ISP to safely realize a much broader range of routing policies without requiring changes to the underlying routers or the BGP protocol itself [103]. Morpheus allows network operators to: (1) make flexible trade-offs between policy objectives through a weighted-sum based decision process, (2) realize customer-specific policies by supporting multiple route-selection processes in parallel, and allowing customers to influence the decision processes, and (3) configure the decision processes through a simple and intuitive configuration interface based on the Analytic Hierarchy Process, a decision-theoretic technique for balancing conflicting objectives. Implemented as an extension to the XORP software router [110], our Morpheus prototype system can support a large number of different policies simultaneously while handling the high rate of BGP updates experienced in large ISPs.

1.3.3 A Technique for Managing Network Changes Without Disruption

Finally, we propose the separation between "physical" and "logical" configurations of routers as a way to solve many network-management problems that involve changes to physical equipment in the network. This abstraction leads us to the design and prototype implementation of "virtual router migration" (VROOM), a new, generic technique that avoids unnecessary changes to the logical topology by allowing (virtual) routers to freely move from one physical node to another [104]. In addition to simplifying existing network-management tasks like planned maintenance and ser-

vice deployment, VROOM can also help tackle emerging challenges such as reducing energy consumption. We present the design, implementation, and evaluation of novel migration techniques for virtual routers with either hardware or software data planes (where packets are forwarded). Our evaluation shows that VROOM is transparent to routing protocols and results in no performance impact on the data traffic when a hardware-based data plane is used.

Collectively, the contributions of this dissertation provide simple and effective systems solutions for an ISP to autonomously provide customized route selection services to its neighbors, and handle a range of existing and emerging network management tasks without disrupting routing protocols or data traffic. And these benefits are achieved provably without affecting the stability of the global Internet.

Chapter 2, 3 and 4 describes NS-BGP, Morpheus and VROOM in detail, respectively. Chapter 5 presents the integrated view of the Morpheus and VROOM systems and concludes the dissertation.

Chapter 2

Neighbor-Specific BGP (NS-BGP): More Flexible Routing Policies While Improving Global Stability

2.1 Introduction

The tens of thousands of independently operated ASes in the Internet have different preferences for the kinds of paths that should carry their traffic. For example, an online gaming provider may prefer paths with low latency, whereas a financial institution may prioritize security over performance. Unfortunately, in today's BGP, each router selects and advertises a *single* best route, limiting an AS's ability to offer customized route selection for its neighbors. As we show in this chapter, with simple extensions to the protocol, a router could offer different interdomain routes to different neighbors. However, greater flexibility in selecting routes should not come at the expense of global stability—a perennial concern with today's routing system. In this chapter, we prove a surprising result: comparing to conventional BGP, *less* restrictive conditions on local routing policies are sufficient to ensure global stability, when an AS is allowed to select different routes

for different neighbors.

2.1.1 A Case for Neighbor-Specific BGP

In today's BGP [80], each router selects a single best route (per destination) and only this route can be announced to its neighbors. Twenty years after BGP was first proposed, this "one-route-fits-all" design has become a frustrating limitation to Internet Service Providers (ISPs) that want to capitalize on their network connectivity by offering customized route selection service to their neighbors. We argue that such flexible route selection (which we dub "neighbor-specific BGP," or "NS-BGP") is beneficial for three main reasons:

- **Many ISPs have rich path diversity.** ISPs offering transit service usually connect to many neighboring ASes, often in multiple locations [70, 68]. For example, ISP Z in Figure 2.6 has four different router-level paths to D, through three different neighboring ASes. Various studies have quantified the rich path diversity seen by large ISPs. For example, at least 2% of all the ASes (which are likely to be tier-1 or tier-2 ISPs) have ten or more unique AS paths for certain destinations [70]. A survey conducted in April 2007 on the NANOG mailing list shows that 5-10 router-level paths per prefix is quite common in large networks, with some prefixes having more than 20 different paths [71]. A detailed study of an individual ISP reported an average of 20 router-level paths for each prefix [97]. These statistics all suggest that large ISPs often have many downstream routes to choose from.

- **Different paths have different properties.** The many alternative routes a large ISP has can have different security and performance properties. In both cases, rich path diversity brings benefits.

  Security: Prefix and sub-prefix hijacking, in which a prefix/sub-prefix is announced by an AS that does not legitimately own it, can cause serious, even disastrous, damage (e.g., in case of online banking) to network users [59]. It was recently shown that path diversity from

a richly connected provider (e.g., tier-1) alone can be very effective in helping its customers resist prefix/sub-prefix hijacks, as it is very hard to hijack all the routes seen by a large ISP [108, 59].

Performance: Path performance (e.g., delay, loss, etc.) is another important factor ISPs should take into account when selecting routes, especially those ISPs that host real-time applications, such as voice-over-IP, video conferencing, or online gaming. However, the current BGP decision process considers little about path performance: the only relevant metric—AS-path length—is a poor indicator of path performance [87, 91, 92]. As a result, alternative BGP paths often have significantly better performance than the default paths [30]. Large ISPs can select better performing paths by leveraging their path diversity [30]. Although some intelligent route control products exist for multi-homed enterprise networks [23], there is no similar counterpart solution in large carrier ISPs.

- **Different neighbors may want different paths.** Different neighbors of an ISP may have very different requirements on the types of routes they want. For example, financial institutions may prefer the most secure paths (e.g., paths that avoid traversing untrusted ASes, such as ASes known to censor traffic), while providers of interactive applications like online gaming and voice over IP may prefer paths with low latency. If such options were available, they might be willing to pay a higher price to have the paths they want. Yet some other neighbors may be perfectly happy with whatever paths the ISP provides for a relatively low price.

  Unfortunately, although large ISPs have the path diversity and strong economic incentive to provide customer-specific routes, they do not have the means to do it today—the BGP decision process selects the same best route for all customers connected at the same edge router, precluding the "win-win" opportunity for large ISPs and their customers.

Ideally, an ISP would be able to offer different routes to different neighbors, regardless of whether they connect to the same edge router. Fortunately, such neighbor-specific route selection is possible without changing the BGP message format or the way neighboring ASes exchange route announcements. As a result, an individual ISP can independently deploy NS-BGP and offer value-added route-selection services. All the changes required for an AS to deploy NS-BGP are within its own network and practically feasible, as discussed in Section 2.5.

2.1.2 Stability Concerns of Greater Flexibility

Despite the benefits of greater flexibility, enhancements to BGP should not come at the expense of global stability. In fact, even *without* neighbor-specific route selection, today's BGP can easily oscillate, depending on the local policies ASes apply in selecting and exporting routes [49, 48]. Over the years, researchers have developed a reasonably good understanding of the trade-offs between local flexibility and global stability [43, 42, 46, 35]. Rather than relying on Internet-wide coordination, researchers searched for practical constraints on local policies that would ensure global stability. In practice, policies are typically constrained by the business relationships between neighboring ASes [43]. For example, a *customer* AS pays its *provider* AS for connectivity to the rest of the Internet, whereas *peer* ASes carry traffic between their respective customers free of charge. These financial arrangements affect how ASes select and export routes, and how new relationships form:

- **Prefer customer routes over peer or provider routes (preference condition):** When selecting a route for a destination, an AS prefers a (revenue-generating) route through a customer over routes through a peer or provider.

- **Export only customer routes to peers or providers (export condition):** An AS can export routes through any neighbor to its customers, but can only export routes through its customers to its peers and providers. That is, an AS provides *transit* services only to its

customers.

- **No cycle of customer-provider relationships (topology condition):** No AS is its own (direct or indirect) provider. That is, the AS-level topology does not contain any cycle of provider-customer edges.

Collectively, these three properties (known as the "Gao-Rexford conditions") ensure the interdomain routing system converges to a stable state without global coordination [43].

The "Gao-Rexford" conditions reflect common business practices in today's Internet, which may explain why the interdomain routing system is generally stable in practice. However, these conditions may be too restrictive for ISPs to offer customized route selection. In particular, ISPs may want to violate the *preference condition* to (1) have different preferences for different neighbors and (2) perhaps even prefer peer or provider routes for some (high-paying) customers. Therefore, we ask the following natural questions: *"Would violating the prefrence condition lead to routing instability in NS-BGP?"* and *"What sufficient conditions (the equivalent of the Gao-Rexford conditions) are appropriate for NS-BGP?"* Answering these questions is crucial to know if customized route selection is possible without sacrificing global stability, and without imposing onerous restrictions on how ASes exploit the extra flexibility.

2.1.3 Relaxing the "Prefer Customer" Condition

In this chapter, we prove that the *more* flexible NS-BGP requires significantly *less* restrictive conditions to guarantee routing stability. Specifically, the "prefer customer" preference condition is no longer needed. Instead, an AS can freely choose *any* "exportable" path (i.e., a path consistent with the export condition) for each neighbor without compromising global stability. That is, an AS can select *any route* for a customer, and *any customer-learned route* for a peer or provider. Intuitively, this is because in NS-BGP, a route announced to a peer or provider is no longer dependent on the presence or absence of any *non-exportable* (e.g., peer- or provider-learned) routes chosen for

customers (as illustrated in the example in Section 2.3.3 and in Figure 2.2).

This condition provides new understanding of the long-believed fundamental trade-off between "local flexibility" and "global stability" in interdomain routing. We make three main contributions in this work:

- An NS-BGP model that captures neighbor-specific route selection and also simplifies the modeling of export policies. (Section 2.2)
- A proof of a sufficient condition for NS-BGP stabiliy that relies only on the export and topology conditions. (Section 2.3)
- Observations that (1) the above NS-BGP stability conditions are robust to failures and other topology changes, (2) NS-BGP can be safely deployed by individual ASes incrementally, (3) compared to BGP, NS-BGP's is less prone to routing anomalies such as "BGP wedgies". (Section 2.4)

We also discuss the practical issues associated with deploying NS-BGP in Section 2.5, including dissemination of alternative routes within an AS, using tunneling to ensure incoming packets (from a neighboring AS or the ISP's own local hosts) traverse the chosen paths, and different models of providing customized route selection. In addition to studying stability issues about NS-BGP, we were also curious about the implications of neighbor-specific route selection on recent theoretical results about the *incentive compatibility* of BGP [67, 44]. We show in Section 2.6 that, as in conventional BGP, rational ASes have an incentive to lie about the paths they are using in NS-BGP. Yet, we argue that this does not affect our positive results regarding NS-BGP stability. Section 2.7 presents related work, and Section 2.8 summarizes the chapter.

2.2 Neighbor-Specific BGP (NS-BGP)

In this section, we formally present Neighbor-Specific BGP (NS-BGP). NS-BGP inherits everything from conventional BGP (from the message format to the way messages are disseminated between ASes) except for the way it selects routes and how messages are disseminated within the AS. We first present a formal model of neighbor-specific route selection, and then define the notion of *stable path assignment* in preparation for the analysis of NS-BGP stability properties in Section 2.3. Finally, we highlight the key novel features of the NS-BGP by contrasting it with conventional BGP.

2.2.1 Preliminaries

In our NS-BGP model, the topology of an interdomain routing system is described as an *AS graph* $G = (V, E)$, where the set of vertices (nodes) V represents the ASes, and the set of edges E represents links between ASes. V consists of n *source nodes* $\{1, \ldots, n\}$ and a special *destination node* d to which all other (source) nodes attempt to establish a path. (This formulation makes sense as routes to different destination ASes/prefixes are computed independently.) E consists of *directed* edges. That is, if nodes u and v have a bi-directional link between them, we have $\{u, v\} \in E$ and $\{v, u\} \in E$, where $\{u, v\}$ is the directed edge from u to v, and $\{v, u\}$ is the directed edge from v to u.

Similar to [48], we define a *path* P in G as either the empty path, denoted by ϵ, or a sequence of nodes $(v_k\ v_{k-1}\ \ldots\ v_0)$, $k \geq 0$, such that for each i, $k \geq i > 0$, $\{v_i, v_i - 1\} \in E$. Each non-empty path $P = (v_k\ v_{k-1}\ \ldots\ v_0)$ has a direction from its *first node* v_k to its *last node* v_0. For each $v \in V$, $\mathcal{P}^v$ denotes the set of *all* simple paths (i.e., paths that do not contain repeated nodes) that has v as the first node and d as the last node, plus the empty path ϵ. If $P = (v\ v_k\ \ldots\ v_1\ d)$ is in $\mathcal{P}^v$, then the node v_k is called the *next hop* of v in path P. For each $\{u, v\} \in E$, $\mathcal{P}^{\{u,v\}}$ denotes the set of *all* simple paths that have $\{u, v\}$ as the first edge (i.e., u as the first node, v as u's next hop) and d as

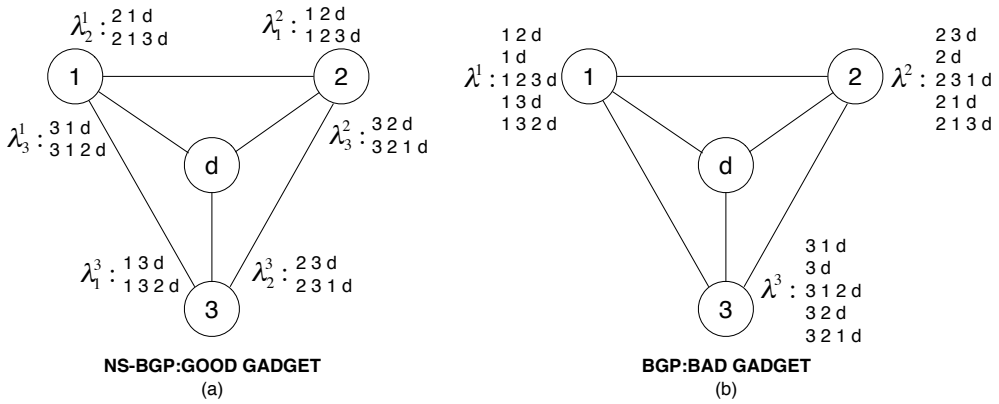


Figure 2.1: NS-BGP vs. BGP: for NS-BGP, ranking function λ_u^v ranks all possible simple paths for edge $\{u, v\}$, or equivalently, for node v's neighbor u (starting from the highest ranked); for BGP, ranking function λ^v ranks all possible simple paths for node v (starting from the highest ranked).

the last node, plus the empty path ϵ. It is easy to see that, for any non-empty path $P \in \mathcal{P}^v$, there is a corresponding path $P' \in \mathcal{P}^{\{u,v\}}$ such that $P' = (u\ v)P$. Here we use $(u\ v)P$ to denote the operation of adding a new first edge $\{u, v\}$ to the path P that starts at node v, so that the new path P' starts at node u, traverses the edge $\{u, v\}$, and then follows path P from v to d. Collectively, we use $\mathcal{P}^{\{u,v\}}$ to denote the set of $P' = (u\ v)P$ for all $P \in \mathcal{P}^v$ and $\{u, v\} \in E$, plus the empty path ϵ.

2.2.2 Neighbor-Specific Route Selection Model

As mentioned in Section 2.1, BGP uses a "one-route-fits-all" route selection model that requires a router to select a single best route for all neighbors. That is, each router uses a single "ranking function" to select the best route for all its neighbors, as shown in Figure 2.1(b). In NS-BGP, we enable customized route selection by allowing a router to select routes on a per neighbor or (equivalently) per edge-link basis. Unlike conventional BGP, this new route selection model allows each router to have *multiple* ranking functions and use a different ranking function to select route for a different neighbor, as shown in Figure 2.1(a). For simplicity, we use "nodes" to denote ASes (instead of routers) in the following model. We discuss the practical issues of how to realize this AS-level route selection model in Section 2.5.

Edge-based ranking functions: In the NS-BGP route selection model, for each edge $\{u, v\}$, there is a *ranking function* λ_u^v, defined over $\mathcal{P}^{\{u,v\}}$, which represents how node v ranks all possible paths for edge $\{u, v\}$ (or equivalently, for neighbor u) to reach d. If $P_1, P_2 \in \mathcal{P}^{\{u,v\}}$ and $\lambda_u^v(P_1) < \lambda_u^v(P_2)$, then P_2 is said to be *preferred over* P_1. We require λ_u^v to impose a strict order (with no ties) over all paths in $\mathcal{P}^{\{u,v\}}$, as v must select a single best path for u.

In NS-BGP, each source node $v \in V$ repeatedly solves the following *route selection problem*, whenever it receives an update of the set of available paths to destination node d:

Definition 1 (Route selection problem)**.** *Given a set of available paths $\mathcal{P}_a^v \subseteq \mathcal{P}^v$ to destination d, choose a best path from $\mathcal{P}_a^{\{u,v\}} = (u, v)\mathcal{P}_a^v$ for each edge $\{u, v\}$ according to the ranking function λ_u^v.*

As the name "Neighbor-Specific BGP" suggests, different edges $\{u, v\}$ and $\{w, v\}$ that point to v from different neighbors u and w can have different ranking functions λ_u^v and λ_w^v, respectively. For example, in Figure 2.1(a), node 1 has two different ranking functions for the two edges $\{2, 1\}$ and $\{3, 1\}$ (or equivalently, for its two neighbors 2 and 3): $\lambda_2^1 = ((2\ 1\ d) > (2\ 1\ 3\ d) > \epsilon)$ (from the most preferred path to the least preferred path), and $\lambda_3^1 = ((3\ 1\ d) > (3\ 1\ 2\ d) > \epsilon)$. Nodes 2 and 3 are similar.

Policy abstraction: Since the set of available paths includes the empty path ($\epsilon \in \mathcal{P}^{\{u,v\}}$), the ranking function λ_u^v can also model v's *export policy* for u (in addition to modeling v's route selection policy for u). This is because if v's export policy does not allow announcing a path P to u, it is equivalent to make P less preferred than the empty path ϵ in the ranking function, i.e., $\lambda_u^v(P) < \lambda_u^v(\epsilon)$. For instance, in Figure 2.1(a), if node d is node 1's customer whereas both nodes 2 and 3 are node 1's peers or providers, node 1 could rank the empty path ϵ higher than all the paths learned from node 3 in λ_2^1 to enforce the "no transit service for peer or provider" export policy, e.g., $\lambda_2^1 = ((2\ 1\ d) > \epsilon > (2\ 1\ 3\ d))$.

2.2.3 Stable Path Assignment

Section 2.2.2 defines the route selection model every *individual* node uses in NS-BGP. We now define the *collective* outcome of the route selection processes run by the individual nodes — the path assignment.

Definition 2 (Path assignment)**.** *An NS-BGP* path assignment *is a function π that maps each edge $\{u, v\} \in E$ to a path $\pi(\{u, v\}) \in \mathcal{P}^{\{u,v\}}$. $\pi(\{u, v\}) = \epsilon$ means that $\{u, v\}$ is not assigned a path to d.*

Definition 3 (Consistent path assignment)**.** *A* consistent path assignment *is a path assignment for which the following statement is true: For each $\{u, v\} \in E$, if $\pi(\{u, v\})$ has $\{v, w\}$ as its second edge (right after $\{u, v\}$), then $\pi(\{u, v\}) = (u, v)\pi(\{v, w\})$.*

Definition 4 (Stable path assignment)**.** *A path assignment π is* stable at edge $\{u, v\}$ *if the following two statements are true: (1) π is a consistent path assignment, (2) For every edge $\{v, w\} \in E$, if $\pi(\{u, v\}) \neq (u, v)\pi(\{v, w\})$, then $\lambda_u^v((u, v)\pi(\{v, w\})) < \lambda_u^v(\{u, v\})$.*

This definition implies that, if a path assignment of edge $\{u, v\}$ is stable, it will not change given any possible available routes. For example, in Figure 2.1(a), a stable path assignment for all edges is shown in Table 2.1.

Table 2.1: An example of stable path assignment for the system shown in Figure 2.1(a)

Edge	Stable path of the edge	Edge	Stable path of the edge
$\{1, d\}$	$(1\ d)$	$\{2, d\}$	$(2\ d)$
$\{3, d\}$	$(3\ d)$	$\{1, 2\}$	$(1\ 2\ d)$
$\{1, 3\}$	$(1\ 3\ d)$	$\{2, 1\}$	$(2\ 1\ d)$
$\{2, 3\}$	$(2\ 3\ d)$	$\{3, 1\}$	$(3\ 1\ d)$
$\{3, 2\}$	$(3\ 2\ d)$		

2.2.4 BGP vs. NS-BGP

Our model differs from the conventional BGP model [48] in the following three respects.

Ranking function(s): node-based *vs.* edge-based: The conventional BGP model requires v to use a single ranking function λ^v for all neighbors, as shown in 2.1(b), offering little flexibility for node v to select the path that best meets an individual neighbor's need. In contrast, the NS-BGP model allows each edge $\{u, v\}$ to have a separate ranking function λ^v_u, which allows v to provide customized route selection for individual neighbors, as shown in Figure 2.1(a).

Path assignment: node-based *vs.* edge-based: In the conventional BGP model, every *node* v gets assigned one path $\pi(u)$. As a result, all of u's neighbors learn the same path from u [1]. Whereas in the NS-BGP model, every *edge* $\{u, v\}$ is assigned a path $\pi(\{u, v\})$. This allows every node u to *simultaneously* utilize up to k paths to forward traffic from its neighbors as well as its own traffic, where k is the number of nodes $v \in V$ such that $\{u, v\} \in E$.

Export policy modeling: separate *vs.* integrated: Although conventional BGP supports per neighbor export policies, it uses a single ranking function λ^v to select routes for all neighbors. As a result, export policies must be modeled *separately* from the route selection process. Such separation is no longer necessary in the NS-BGP model, as node v's export policy for neighbor u can be conveniently incorporated in the ranking function λ^v_u. For example, if u is v's peer or provider, in the ranking function λ^v_u, v can simply rank the empty path ϵ higher than all peer- or provider-learned paths to implement the "no transit service for peer or provider" export policy.

[1] In practice, an AS usually consists of multiple routers, each of which may learn different paths. Thus, neighbors connect to the AS at *different* edge routers might learn different paths, due to "hot potato routing". Nevertheless, NS-BGP provides a far more flexible and systematic way for ASes to provide customized route-selection service, independent of whether neighbors connect to the same edge router or not.

2.3 Sufficient Conditions for NS-BGP Stability

The "Gao-Rexford" conditions [43] state that, if all ASes follow the export, preference, and topology conditions, BGP is guaranteed to converge to a stable state. Fortunately, we find that much *less* restrictive conditions are sufficient to guarantee convergence under the *more flexible* NS-BGP. Specifically, the "prefer customer" condition is no longer needed in NS-BGP—individual ASes can freely choose *any* "exportable" routes without compromising global stability. In this section, we first define the notion of *NS-BGP safety*, which implies that an NS-BGP routing system always converges to a stable path assignment. We then review *Iterated Dominance* (presented in [86]), the machinery we use in our proof. We next present simple examples that illustrate why NS-BGP requires less restrictive conditions for safety than conventional BGP, before presenting the proof of our safety result.

2.3.1 Formal Definition of NS-BGP Safety

For any policy-based (non-shortest-path) routing protocol (such as BGP or NS-BGP), *safety* is a top concern, as persistent route oscillations can significantly impact end-to-end performance, and even threaten the reachability of network destinations. BGP *safety* can be loosely defined as a routing system that always converges to a "stable" state. Recall that a stable state is a path assignment that does not change given any possible route announcements. Thus, once a system is in a stable state, it will never experience any further changes (provided the network topology and every node's routing policy remain the same). To formally define NS-BGP safety, we first need to introduce the notion of "AS activation sequences".

AS activation sequences: As in conventional BGP, the routing outcome of NS-BGP is built, hop-by-hop, as knowledge about how to reach a destination d propagates throughout the network. The process begins when d announces itself to its neighbors by sending update messages. From this

moment forward, every node v repeatedly picks a path for each edge $\{u, v\} \in E$, based on the most recent updates of routes to d it received from its neighbors. As in [49, 48], the network is assumed to be *asynchronous*. That is, edges can be *activated* (i.e., get assigned new paths) at different times, and update messages can be delayed or even lost (as long as they are retransmitted eventually). We refer readers to [48] for a thorough explanation of this asynchronous environment.

Definition 5. *An NS-BGP routing system is safe if it always converges to a stable path assignment from any initial path assignment, and for any AS activation sequence.*

2.3.2 Iterated Dominance

It was observed in [86] that all known conditions that guarantee the safety of conventional BGP (e.g., "No Dispute Wheel" [48] and the "Gao-Rexford" conditions [43]) share a common structure [86], referred to as "*Iterated Dominance*". This property is related to the notion of dominance-solvability in game theory [76]. Iterated Dominance is an underlying structure of a routing instance, which will enable us to show that, for any activation sequence, NS-BGP is bound to converge to a *unique* stable state. Informally, Iterated Dominance means that, as time advances, nodes' feasible choices of routes gradually become more and more limited, until eventually every node's route is fixed. Thus, Iterated Dominance provides us the means to present a *constructive*, and general, proof for NS-BGP safety.

We shall later show that the commercial setting considered in this chapter is simply a special case of Iterated Dominance. To define Iterated Dominance, we first require the following definitions:

Definition 6 (Consistent paths I)**.** *We say two paths P_1 and P_2 are* consistent *if the following statement holds: For every edge $\{i, j\}$ that is on both P_1 and P_2, the suffix of P_1 that leads from j to d is identical to the suffix of P_2 that leads from j to d. In addition, two paths are also consistent if they that do not share any common edge.*

Definition 7 (Consistent paths II)**.** *Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a set of paths in G. We say that a path Q in G is consistent with $\mathcal{P}$ if it is consistent with every path in $\mathcal{P}$.*

Definition 8 (Feasible paths)**.** *Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a set of paths in G. We define the set of* feasible *paths $\mathcal{Q}$* given *$\mathcal{P}$ to be the set of all paths in G that are consistent with $\mathcal{P}$.*

Definition 9 (Iterated Dominance)**.** *We say that* Iterated Dominance *holds if there exists an order over all edges in G: $e_1, \ldots, e_{|E|}$ ($e_i \in E$, $1 \leq i \leq |E|$), for which the following three statements hold:*

- *There exists a set of paths $P_{e_1}, \ldots, P_{e_{|E|}}$ such that for every $1 \leq i \leq |E|$, P_{e_i} is a path to d that has e_i as the first edge.*
- *For every $1 \leq i \leq |E|$, $P_{e_i} = e_i P_{e_k}$ for some $0 \leq k < i$. (We define e_0 to be the empty path ϵ).*
- *For every $1 \leq i \leq |E|$, P_{e_i} is e_i's most preferred path in the set of feasible path given $\{P_{e_1}, \ldots, P_{e_{|E|}}\}$.*

Intuitively, this definition means that once the paths assigned to edges that come before a certain edge are fixed, that edge's path is its most preferred feasible path. Iterated Dominance has the nice property that, if it exists in a routing system, it trivially and intuitively induces convergence to a stable path assignment.

Proposition 2.3.1. *If Iterated Dominance holds for an interdomain routing instance, then NS-BGP is safe for that routing instance. Moreover, NS-BGP always converges to a unique stable path assignment.*

Proof. The proof immediately follows from the Iterated Dominance property. If Iterated Dominance holds then there must be an order over the edges $e_1, \ldots, e_{|E|}$ such that, for every $1 \leq k \leq |E|$ an edge e_k can be assigned its most preferred feasible path (given that $e_1, \ldots, e_{k-1}$ are assigned

$P_{e_1}, \ldots, P_{e_k}$), regardless of what paths are assigned to $e_{k+1}, \ldots, e_{|E|}$. Thus, we can simulate the execution of an activation sequence of NS-BGP, which shows that the routing system must converge to a unique stable path assignment:

At some point in time e_1 will learn of its most preferred path P_{e_1}. From that moment forward, e_1 will stick to the path P_{e_1} (which, by the definition of Iterated Dominance, is always available to e_1). Now, consider e_2. Once e_1's path is fixed, by the definition of Iterated Dominance, e_2 can get its most preferred feasible path P_{e_2}. Therefore, from some moment in time onwards (when an update message containing P_{e_2} reaches e_2), e_2's path will be fixed and never change. By definition of Iterated Dominance, we can continue iteratively fixing other edges' paths until every edge has a fixed path. Observe that the resulting path assignment is stable, because after each edge e_i gets its path P_{e_i}, it will never switch to other paths. □

2.3.3 Examples of Safe NS-BGP Systems

Before presenting the formal proof of our main result, we first use an example to illustrate why safety might be easier to achieve for NS-BGP than for conventional BGP. Figure 2.1(b) shows a routing system in which BGP will always diverge, which is called BGP BAD GADGET [48]. In this example, λ^1, λ^2 and λ^3 are the ranking functions of nodes 1, 2 and 3, respectively. It is easy to construct an activation sequence (presented as a sequence of path assignments) according to the ranking functions, for example: ($(1\ d)$, $(2\ d)$, $(3\ d)$) → (underline($(1\ 2\ d)$), $(2\ d)$, $(3\ d)$) → boxed((1 2 d), underline(2 3 d), (3 d)) → (underline($(1\ d)$), $(2\ 3\ d)$, $(3\ d)$) → ($(1\ d)$, $(2\ 3\ d)$, underline($(3\ 1\ d)$)) → ($(1\ d)$, underline($(2\ d)$), $(3\ 1\ d)$) → (underline($(1\ 2\ d)$), $(2\ d)$, $(3\ 1\ d)$) → ($(1\ 2\ d)$, $(2\ d)$, underline($(3\ d)$)) → boxed((1 2 d), underline(2 3 d), (3 d)). (An underlined path indicates that it has changed from the previous path assignment.) Notice that the third path assignment is the same as the last path assignment. Therefore, the system will continue to oscillate and never terminate.

To see how NS-BGP can help in cases like this, we transformed the BGP routing system in Figure 2.1(b) to an "equivalent" NS-BGP system in Figure 2.1(a). This is an "extreme" example in that we assume every node is willing to select paths for each incoming edge (i.e., each neighbor)

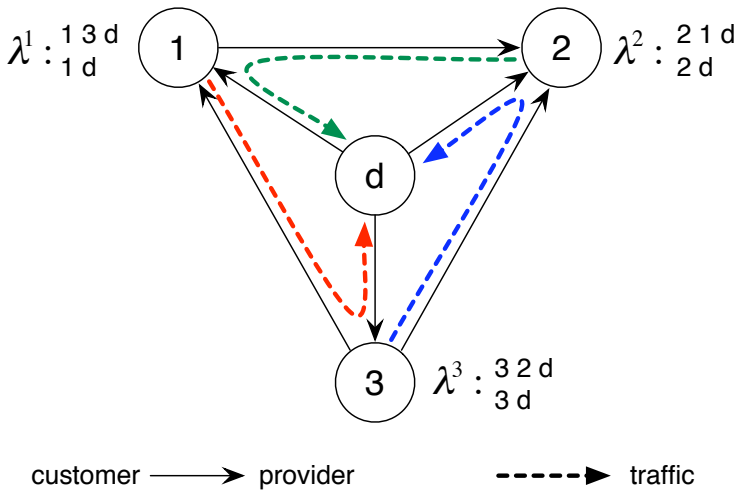


Figure 2.2: Why NS-BGP does not need the "preference condition" and can safely allow nodes to choose any exportable routes: the dotted arrows denote the stable path assignment, in which every node i ($i = 1, 2, 3$) makes the direct path $\{i, d\}$ available to its clockwise neighbor while using a different path itself.

completely according to the edge's (or equivalently, the neighbor's) ranking function. For example, when selecting best path for edge $\{2, 1\}$, node 1 in Figure 2.1(a) uses a ranking function λ_2^1 that is essentially the same as node 2's ranking function λ^2 in Figure 2.1(b). The only difference is that, since λ_2^1 is defined over $P^{\{2,1\}}$ whereas λ^2 is defined over P^2, only a subset of the paths in P^2 that begin with edge $\{2, 1\}$ (e.g., $(2\ 1\ d)$ and $(2\ 1\ 3\ d)$) are included in λ_2^1. We omit the empty path ϵ for simplicity. It is easy to see that the transformed BGP `BAD GADGET` in Figure 2.1(a) becomes an NS-BGP `GOOD GADGET`, i.e., a routing system in which NS-BGP will always converge to a unique stable path assignment. In this case, the unique stable path assignment for all edges is: ($(1\ d)$, $(2\ d)$, $(3\ d)$, $(1\ 2\ d)$, $(1\ 3\ d)$, $(2\ 1\ d)$, $(2\ 3\ d)$, $(3\ 1\ d)$, $(3\ 2\ d)$).

This example illustrates why safety might be easier to obtain for NS-BGP than for conventional BGP. In practice, however, relying on such completely "selfless" routing policies is unrealistic. This prompts us to investigate the safety conditions for NS-BGP in a more realistic commercial setting that accounts for the business relationships between ASes. For example, consider Figure 2.2, where node d is a customer of nodes 1, 2 and 3. Node 3 is a customer of nodes 1 and

2, and node 1 is a customer of node 2. It is easy to see there is no "customer-provider" cycle in the graph so the topology condition holds. We also require nodes 1, 2 and 3 to adhere to the export condition and export only customer routes to peers or providers. Now we compare BGP and NS-BGP and analyze why the "prefer customer" condition is necessary in conventional BGP but redundant in NS-BGP. First, note that the ranking function λ_3 prefers provider-learned route $(3\ 2\ d)$ over customer-learned route $(3\ d)$, violating the preference condition for the regular BGP. As a result, (not surprisingly) the routing system is a BGP `BAD GADGET`.

A key observation about the instability of the BGP system in Figure 2.2 is that the *availability* of route $(1\ 3\ d)$ to node 1 is dependent upon the *unavailability* of route $(3\ 2\ d)$ to node 3—if route $(3\ 2\ d)$ is available to 3, it will choose route $(3\ 2\ d)$ over $(3\ d)$, and announce no route to node 1; whereas if route $(3\ 2\ d)$ is not available to 3, it will choose route $(3\ d)$ and announce it to node 1 (since $(3\ d)$ is a customer-learned route). Things work differently in NS-BGP. NS-BGP ensures that a route announced to a peer or provider does not change based on the presence or absence of any *non-exportable* (e.g., peer- or provider-learned) routes. That is, in this example, node 3 learning $(3\ 2\ d)$ (a provider-learned route) should not affect whether node 3 exports $(3\ d)$ to node 1 (which is also a provider). Fundamentally, this is because, in NS-BGP, node 3 can announce a different route $(3\ d)$ to node 1 than the route it selects for its own traffic, namely $(3\ 2\ d)$.

2.3.4 Safety Conditions for NS-BGP

To prepare for our analysis, we first define some terminology: We say that an edge $e = \{u, v\} \in E$ is a *customer edge* if v is u's customer. Similarly, we say that an edge $e = \{u, v\}$ is a *peer edge* or a *provider edge* if v is u's peer or provider, respectively. Observe that the "No customer-provider cycle" topology condition in the "Gao-Rexford" guidelines can now be interpreted as stating that there must be no cycles in the graph containing only customer edges or only provider edges. Also observe that the "Export only customer routes to peer or providers" condition means that if a path

P contains a customer edge or a peer edge, then all edges that come after that edge (towards the destination) must also be customer edges, allowing us to simply disregard all other types of paths in our analysis.

Lemma 2.3.2. *If the Topology and Export conditions hold for an NS-BGP routing instance, then Iterated Dominance holds for that routing instance.*

Proof. We shall show that an order over edges $e_1, ..., e_{|E|}$, as in the definition of Iterated Dominance, exists. Obviously, we can set e_1 to be any edge of the form $\{u, d\}$ ($\{u, d\} \in E$) as $(u\ d)$ is the only path that edge has to d. So by setting $P_{e_1} = (u\ d)$, we have found an edge e_1 that fits the definition of Iterated Dominance. The rest of the proof shows how to prove the existence of an edge e_2, as required by the definition of Iterated Dominance. The same method can then be applied recursively to find $e_3, \ldots, e_{|E|}$ (thus concluding the proof).

If there is another edge of the form $\{u, d\}$, we can now set e_2 to be that edge for the same reason as before. We shall now show how to find e_2 as in the definition of Iterated Dominance, if this is not the case. Informally, the proof shall now proceed by iteratively applying the following procedure: Fix an edge e. Go over its most preferred feasible route (given P_{e_1}) until reaching the edge before last, l_1. If edge l_1 fits the description of e_2 then we are done. Otherwise, we apply the same procedure to l_1, moving to the edge before last on l_1's most preferred feasible path, called l_2 (which we regard as a new candidate to be e_2). Thus, we create a sequence of edges $l_1, l_2, \ldots$. We show that this procedure eventually reaches an edge that fits the description of e_2 (thus concluding the proof), because otherwise the "No customer-provider cycle" will be violated (a contradiction).

Formally: Let $e \neq e_1$ be some arbitrarily chosen edge. Let P_e be e's most preferred path among all feasible paths given P_{e_1}. For ease of exposition, we first consider the case in which e is a customer edge.

Now, to find e_2, we shall construct a series of edges $l_1, \ldots, l_k, \ldots$ in the following manner: Let $(i\ j\ d)$ be the two-edge suffix of P_e (i.e., the last two edges on P_e are $\{i, j\}$ followed by $\{j, d\}$). We set l_1 to be $\{i, j\}$. If l_1 prefers $(i\ j\ d)$ over all other feasible paths, then we can set e_2 to be l_1

and P_{e_2} to be $(i\ j\ d)$ (and are done). If, however, l_1's most preferred feasible path P_{l_1} is not $(i\ j\ d)$, we then consider the two-edge suffix of P_{l_1} and set l_2 to be the first of these two edges. For l_2, we repeat the same process we went through for l_1. That is, either l_2 prefers the two-edge suffix of l_1 over any other feasible path (in which case we set e_2 to be l_1, and are done), or we move on to l_3 (which is the first edge of l_2's most preferred path's two-edge suffix). We continue this process, constructing a series of edges $l_1, \ldots, l_k, \ldots$. If this process terminates then we must have reached an edge that fits the description of e_2.

We prove that this process must terminate by showing that if it does not terminate, we will reach a contradiction to the topology condition ("No customer-provider cycles").

First, observe that for any edge l_j in the series of edges $l_1, \ldots, l_k, \ldots$, there exists a path between l_j and l_{j+1} that consists only of customer edges. To see why this is true, consider l_1. We assumed that e was a customer edge. Therefore, by the export condition, any path assigned to e must only consist of customer edges. Since l_1 is on such a path, it must be a customer edge. Using the same argument, we know that l_1 can only be assigned paths consisting of only customer edges. Since l_2 is, by definition, on such a path (l_1's most preferred feasible path), we have shown that the path between l_1 and l_2 consists of customer edges only, so the claim holds for l_1. We can now repeat the same argument for l_2, l_3, etc.

Now, if the process does not terminate, then, since the number of edges is finite, some edge l_i will eventually appear twice in the sequence $l_1, \ldots, l_k, \ldots$. Consider the subsequence of $l_i, \ldots, l_i$ (between l_i's first and second appearance). Because any two consecutive edges in this cyclic sequence have a path between them that consists of only customer-edges, there must exist a customer-provider cycle (i.e., a cycle of only customer edges).

The cases in which e is a peer edge or a provider edge are handled similarly: If e is a peer edge then the edge that comes after it must be a customer edge, so the same arguments as before apply. If e is a provider edge then the process described before will either go through a customer edge or a peer edge (in which case, once again, the arguments above apply) or lead to a cycle of provider

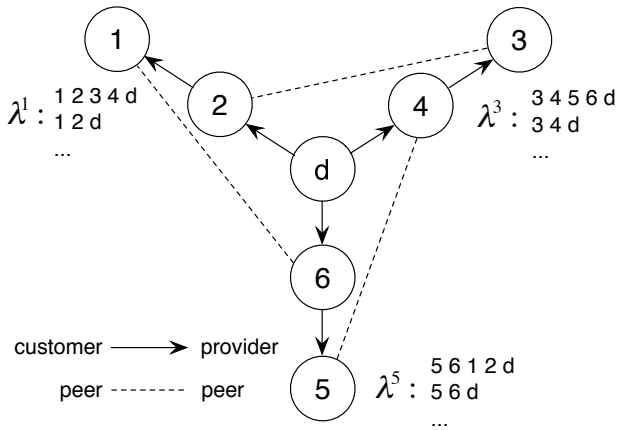


Figure 2.3: Tightness of the NS-BGP safety conditions

edges. □

We are now ready to prove the safety conditions of NS-BGP:

Theorem 2.3.3 (Safety Conditions of NS-BGP)**.** *If the Topology and Export conditions hold then NS-BGP is safe. Moreover, NS-BGP always converges to a unique stable path assignment.*

Proof. Lemma 2.3.2 shows that the topology and export conditions are sufficient to guarantee Iterated Dominance. Therefore, by Proposition 2.3.1, NS-BGP is safe, and always converges to a unique stable path assignment. □

2.3.5 Tightness of the Safety Conditions

In this subsection we show that our NS-BGP safety conditions are "tight", in the sense that a relaxation of either the topology condition or the export condition might result in persistent NS-BGP oscillations.

Consider the example depicted in Figure 2.3. This example can be viewed as an adaptation of the well-known BGP `BAD GADGET` instance described in [48] and Figure 2.1(b) to the neighbor-specific BGP setting. The top two preferred paths in edges $\{1, 2\}$'s, $\{3, 4\}$'s, and $\{5, 6\}$'s ranking functions are listed (from top to bottom) in the figure. We omit the rest of the paths in the ranking

functions for simplicity, as they play no roles in this example. The business relationships between the ASes are described in the figure (where the arrows point from customers to their providers). Observe that the topology condition holds as there are no customer-provider cycles. If we assume that the export constraint also holds then, by Theorem 2.3.3, this NS-BGP routing system is guaranteed to converge to a unique stable path assignment.

What happens if the export condition is removed (i.e., not followed)? We claim that the system will then have no stable path assignment and so will oscillate indefinitely. Observe that if node 2 follows the export condition, it cannot export path $(2\ 3\ 4\ d)$ to node 1, making path $(1\ 2\ 3\ 4\ d)$ unavailable to node 1. Similarly, paths $(3\ 4\ 5\ 6\ d)$ and $(5\ 6\ 1\ 2\ d)$ are not available to nodes 3 and 5, respectively. But if the export condition is not followed, these paths will become available. Assume, to lead to a contradiction, that a stable path assignment exists when the export condition is removed. Observe that edge $\{1, 2\}$ must either get the path $(1\ 2\ d)$ or $(1\ 2\ 3\ 4\ d)$ in this path assignment (as it will not settle for a less preferred path than its second preferred path $(1\ 2\ d)$ that is always available). Let us first consider the possibility that $\{1, 2\}$'s path in this stable assignment is $(1\ 2\ d)$. If that is the case, then $\{5, 6\}$ must be getting the path $(5\ 6\ 1\ 2\ d)$. This means that node 5 will not announce $(5\ 6\ d)$ to node 4 (because node 6 announces $(6\ 1\ 2\ d)$, rather than $(6\ d)$, to node 5). Therefore, edge $\{3, 4\}$ is assigned the path $(3\ 4\ d)$, which, in turn, means that edge $\{1, 2\}$ can get its most preferred path $(1\ 2\ 3\ 4\ d)$. Now we have a contradiction—edge $\{1, 2\}$ has an available path $(1\ 2\ 3\ 4\ d)$ which it prefers over the path it is assigned in the stable path assignment $(1\ 2\ d)$. Observe that if, instead, we assume that edge $\{1, 2\}$ gets path $(1\ 2\ 3\ 4\ d)$ in the stable path assignment, then edge $\{3, 4\}$ must get path $(3\ 4\ d)$ in the stable path assignment. We can continue this inference process like above and eventually reach a similar contradiction to edge $\{1, 2\}$'s assigned path.

We have shown that without the export condition, not only is NS-BGP safety not guaranteed but there might not even be a stable path assignment to which it can converge. We make the observation that this is also the case if we remove the topology condition (while leaving the export condition

alone). Consider the same example, only with the following business relationship changes: make nodes 3, 5, and 1 customers of nodes 2, 4, and 6, respectively. Observe that the topology condition no longer holds as we now have a customer-provider cycle ($3 \rightarrow 2 \rightarrow 1 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 3$). Also observe that paths $(1\ 2\ 3\ 4\ d)$, $(3\ 4\ 5\ 6\ d)$, and $(5\ 6\ 1\ 2\ d)$ are now *allowed* by the export condition as a result of the changes in the business relationships we made. Therefore, we can use the same analysis as above to show that no stable path assignment exists if the topology condition is removed.

2.4 Practical Implications

In this section we discuss three practical implications of the NS-BGP safety conditions presented in Section 2.3. Specifically, we show that:

1. Our NS-BGP safety conditions are robust, in the sense that they hold even in the presence of topology changes (e.g., the addition and removal of nodes and/or links due to new business contracts, creation, merger, or disappearance of ASes, network failures, etc.).

2. It is *safe* to deploy NS-BGP incrementally. Global routing stability is guaranteed even if only some of the ASes run NS-BGP, while others continue to run BGP. Moreover, the global routing system is still guaranteed to converge to a *unique* stable path assignment.

3. By allowing arbitrary ranking of exportable paths, NS-BGP naturally supports the important class of "backup" business relationships (i.e., an AS having a backup provider) and is less prone to "Wedgies" [45] than conventional BGP.

   Our NS-BGP safety conditions also provide useful guidance for solving the stability problems of internal BGP (iBGP) within an AS.

2.4.1 Safe Under Topology Changes

We have shown is Section 2.3.4 (Theorem 2.3.3) that if the topology and export conditions hold for a routing instance, then NS-BGP is guaranteed to converge to a stable path assignment. However, does this result still hold in the presence of topology changes? We make the observation that our NS-BGP safety conditions *are* robust in the presence of topology changes.

We first consider topology changes that result in the *removals* of edges and/or vertices from the graph G in our model. Such changes can happen due to network failures (e.g., equipment failures, fiber cuts) or business relationship changes (e.g., termination of a existing BGP peering relationship). We observe that, if the topology condition and the export condition hold for a routing instance, they cannot be violated by removing edges and/or vertices from the network. Hence, after the removal of certain edges and/or vertices, we will end up with a new routing instance for which these two conditions still hold. By Theorem 2.3.3, NS-BGP safety of the new routing instance is guaranteed.

Similarly, when there are topology changes that result in the *additions* of edges and/or vertices from the graph G in our model (e.g., due to the establishment of a new AS or a new BGP peering relationship), we note that our proof of Theorem 2.3.3 still holds for the new routing instance after the topology changes, as long as they do not violate the topology and export conditions. That is, the new vertices and/or edges do not create "customer-provider" cycles and they follow the "export only customer routes to peer or provider" export policy. Since ASes have economic incentive to follow the two conditions, the new routing instance is guaranteed to remain safe.

2.4.2 Safe in Partial Deployment

The proof of the NS-BGP safety conditions in Section 2.3 assumes all ASes in the network run NS-BGP, i.e., a *full deployment* of NS-BGP. However, the actual deployment of NS-BGP will certainly start *incrementally*, as any AS that has deployed NS-BGP individually can immediately start

offering customized route-selection services without collaboration. Therefore, a natural question is whether the NS-BGP safety conditions still hold in a *partial deployment* scenario (with some "early adopter" ASes running NS-BGP, while other ASes still running conventional BGP)?

As we shall now show, the answer to this question is *YES*. That is, NS-BGP can be (under reasonable and realistic assumptions) *incrementally- and partially-deployed* without causing persistent protocol oscillations. We observe that, using the exact same techniques we have used to prove Theorem 2.3.3, we can actually prove a much more general result [2]:

Theorem 2.4.1. *If topology and export conditions hold for a routing system, then, even if some ASes are running NS-BGP while other ASes are still running BGP, as long as the preference condition applies to* the ASes running conventional BGP *(it is not needed for ASes running NS-BGP), the routing system will always converge to a unique stable path assignment.*

That is, as long as the ASes *not running NS-BGP* prefer customer routes to other routes in their route selection, the system will remain safe. We note that this result holds true *regardless* of the number of ASes that are not running NS-BGP, and *regardless* of the locations of these ASes in the network. This result therefore generalizes both Theorem 2.3.3 (which considers cases in which *all* ASes are running NS-BGP) and the "Gao-Rexford" conditions [43] (which apply to cases in which *all* ASes are executing BGP).

We also observe that, by the same arguments as in Section 2.4.1 and 2.4.3, the above safety conditions of a partial NS-BGP deployment still hold in the presence of network topology changes, and a routing system with even partially deployed NS-BGP may experience fewer BGP Wedgies.

2.4.3 Safer With Backup Relationship

As we know, if all ASes follow the "Gao-Rexford" conditions, a BGP routing system is guaranteed to be stable. However, the "Gao-Rexford" conditions only apply to routing systems with

[2] We omit the details of the proof as it follows similar lines of the proof in Section 2.3.4.

the two most common business relationships ("customer-provider" and "peer-peer"). Yet, it has been increasingly common for ASes to establish a third class of business relationships—"backup" relationships—to prevent the loss of network connectivity after a failure. The introduction of backup relationships can cause a routing system to have two stable states (i.e., two stable path assignments), and result in a type of routing anomaly known as a *BGP Wedgie* [45]. We first recall the notion of BGP Wedgies, and then explain why backup relationships in an NS-BGP routing system are less likely to result in BGP Wedgies.

BGP Wedgies: The term "BGP Wedgies", coined in [45], refers to the following problem with BGP: It is common for an AS to have two (or more) upstream providers to avoid a single point of failure in network connectivity. In such cases, the AS usually places a relative preference on the two links its providers use to reach it: one link is defined as the "primary" (preferred), while the other one is defined as the "backup" link. A backup link is intended to be used only when the primary link is temporarily unavailable, therefore is typically much less well-provisioned in terms of bandwidth. It is expected that once the primary link is restored, all traffic should switch back from the backup link to the primary link. BGP Wedgies are anomalous situations in which, even after a failed primary link is restored, the BGP state of the routing system does not "flip back" to the intended state that existed before the link failure.

Consider the example of a Wedgie in conventional BGP, as shown in Figure 2.4. AS d is a customer of ASes 1 and 3, AS 1 is a customer of AS 2, and ASes 2 and 3 are peers. AS d chooses to use the link $\{d, 3\}$ as the primary link and the link $\{d, 1\}$ as the backup link. AS d instructs AS 1 to use path $(1\ d)$ only when there is no other path available (e.g., using the BGP community attribute to mark the path $(1\ d)$ as "backup only" in its route updates). Assume that the original BGP state is such that all ASes are forwarding their traffic to AS d along the path $(1\ 2\ 3\ d)$. Observe that this state is stable (as AS 1 does not announce path $(1\ d)$ to AS 2 when path $(1\ 2\ 3\ d)$ is available). Now, assume that link $\{3, d\}$ goes down for some reason. Since the path $(1\ 2\ 3\ d)$ is

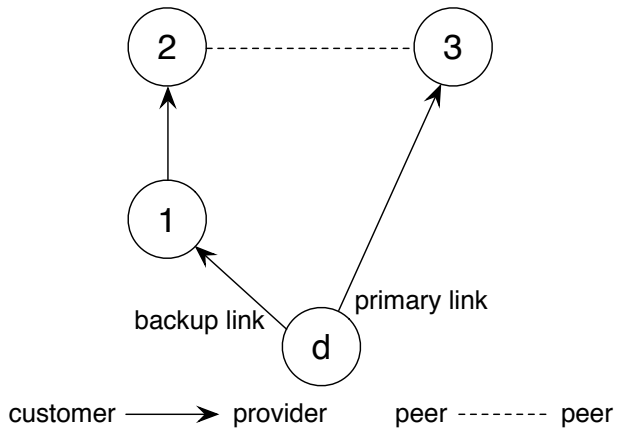


Figure 2.4: A BGP Wedgie: AS 2 will not switch back to path $(2\ 3\ d)$ after the primary link $\{3, d\}$ is restored from a failure.

no longer available, AS 1 will announce path $(1\ d)$ to AS 2, which will in turn announce it to AS 3. In the end, traffic to AS d is forwarded along the path $(3\ 2\ 1\ d)$. Once link $\{d, 3\}$ is restored, a BGP Wedgie occurs: although AS 3 will announce path $(3\ d)$ is available again, AS 2 will not switch back from its current customer-learned path $(2\ 1\ d)$ to a less preferred peer-learned path $(2\ 3\ d)$, and will not announce the path $(2\ 3\ d)$ to AS 1. As a result, AS 1 (and 2) will keep using the backup link even though the primary link has become available again.

NS-BGP helps prevent Wedgies: Let us revisit the example described above. Notice that the Wedgie example in Figure 2.4 will *not* occur if the routing system runs NS-BGP, because AS 2 will have AS 1's ranking function (in this case, $\lambda_1^2 = ((1\ 2\ 3\ d) > \epsilon$), and selects a path for AS 1 on its behalf. So when link $\{d, 3\}$ is restored, AS 2 will learn the path $(3\ d)$ from AS 3 again and announce the path $(2\ 3\ d)$ to AS 1 because $(1\ 2\ 3\ d)$ is 1's most preferred path. Once AS 1 learns this path, it will withdraw the backup path $(1\ d)$ from AS 2 and AS 2 will switch back to use $(2\ 3\ d)$. Therefore, the system will be restored to the original state that existed before the link failure.

As we have seen, NS-BGP prevents Wedgies in certain cases that would have been a problem under conventional BGP. However, NS-BGP is not totally immune to Wedgies. To see this, con-

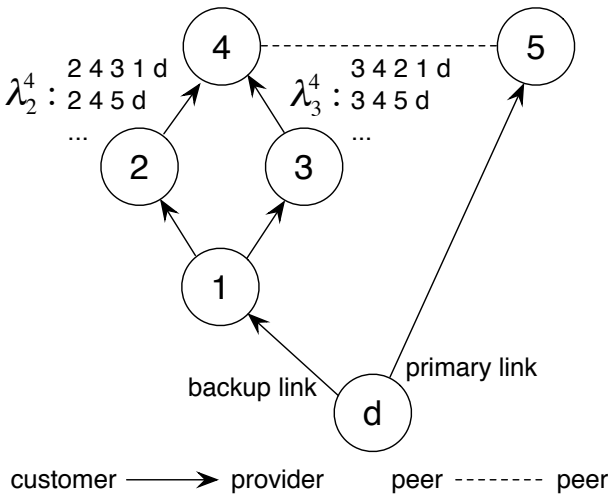


Figure 2.5: An NS-BGP Wedgie: ASes 2 and 3 will not switch back to the path through the primary link $\{5, d\}$ after it is restored from a failure.

sider the example in Figure 2.5. Assume that ASes 2 and 3 make their preferences known to their provider AS 4. In the normal case, all the ASes send their traffic through link $\{5, d\}$. If $\{5, d\}$ fails, then all ASes send traffic through $\{1, d\}$. After $\{5, d\}$ is restored, AS 4 will learn path $(4\ 5\ d)$ from AS 5. But it will not announce the this path to neither 2 or 3, because it has previously announced more preferred paths to AS 2 (path $(4\ 3\ 1\ d)$) and AS 3 (path $(4\ 2\ 1\ d)$). Hence, AS 1 will never learn of the restoration of $\{5, d\}$ and therefore will never withdraw the path $(1\ d)$. This results in a Wedgie.

2.4.4 Preventing Instability in Internal BGP

We note that our NS-BGP safety results, while primarily addressing economic- and engineering-related issues in *interdomain* routing, also have implications for routing *within* an AS. In practice, an AS is itself a complex network consisting of multiple routers in different geographic locations. In backbone networks, these routers exchange routing information using a variant of BGP known as *internal BGP* (iBGP). Since having an iBGP session between each pair of routers does not scale, most large backbones use *route reflectors* or *confederations* to impose a hierarchy for distributing

BGP routing information. A router configured as a route reflector selects a best route on behalf of its client routers, obviating the need for the clients to maintain so many iBGP sessions or learn so many BGP routes. However, previous work has shown that persistent route oscillation can easily occur inside an AS [10, 51, 50], due to the complex interaction of iBGP and Interior Gateway Protocols (IGPs) like OSPF and IS-IS.

The dissemination of routes between route reflectors, and between route reflectors and their clients, parallels the business relationships between ASes in interdomain routing [51]. In particular, the relationship between a route reflector and its clients in iBGP is much the same as the relationship between a provider AS and its customer ASes; similarly, the relationship between two route reflectors is much the same as the relationship between peer ASes in interdomain routing. Depending on how the routers in the AS "rank" the routes they've learned, oscillations can result. In fact, a solution to this problem is to impose a "prefer route-reflector client" condition [51], analogous to the "prefer customer" Gao-Rexford condition. (In practice, this imposes strict restrictions on the IGP configuration, to ensure that route reflectors are topologically "close" to their clients.) Our results for NS-BGP suggest another, more flexible, solution—allow route reflectors to announce different routes to different iBGP neighbors. In particular, a route reflector could disseminate any client-learned route (such as the client-learned route with the closest egress point) to its route-reflector peers, and any route (such as the route with the closest egress point) to its route-reflector clients. This small modification to iBGP would provably ensure iBGP convergence without imposing any restrictions on the IGP configuration.

2.5 Deployment Issues

In this section, we discuss the implementation issues in deploying NS-BGP in practice. First, we describe how an AS can correctly forward traffic from different neighbors (and from within its own network) along different paths. We then discuss how to disseminate multiple routes to the

edge routers of an AS to enable flexible route selection. Finally, we present three models an NS-BGP-enabled AS can use to provide different levels of customized route-selection services. When deploying NS-BGP, an AS can handle all these issues by itself without requiring any changes from neighboring ASes, as no BGP message format or external BGP (eBGP) configuration are needed.

2.5.1 Neighbor-Specific Forwarding

NS-BGP requires routers to be able to forward traffic from different neighbors along different paths. Fortunately, today's routers already provide such capabilities. For example, the "virtual routing and forwarding (VRF)" feature commonly used for Multi-protocol Label Switching Virtual Private Networks (MPLS-VPNs) supports the installation of different forwarding-table entries for different neighbors [77].

Since an AS typically consists of many routers, traffic entering from various *ingress* routers of the AS must be forwarded to the correct *egress* routers. In conventional BGP, this is achieved in a hop-by-hop fashion to ensure that all routers in the AS agree to forward traffic to the closest egress point that has one of potentially multiple "equally good" best paths to the destination. For example, in Figure 2.6, if $R5$ learns one path from $R3$ and another path from $R4$ to D, and the two routes are considered "equally good" in BGP's route-selection process, it will choose to use the closest egress point (according to the IGP distances). However, this approach no longer works in NS-BGP, as traffic entering the AS at the same ingress point may be from different neighbors (ingress links), and thus may need to be forwarded to different egress points, or different egress links of the same egress point. Fortunately, ASes have an efficient solution available—encapsulation (or tunneling). Many commercial routers deployed in today's networks can perform MPLS or IP-in-IP encapsulation/decapsulation at line rate. To provide customized forwarding for neighbors connected at the same edge router, the tunnels need to be configured from ingress *links* (rather than ingress routers) to egress *links* (rather than egress routers). For example, in Figure 2.6, $C1$'s and $C2$'s traffic can be tunneled from $R1$ to $R6$ and $R7$ (that connect to the same egress point

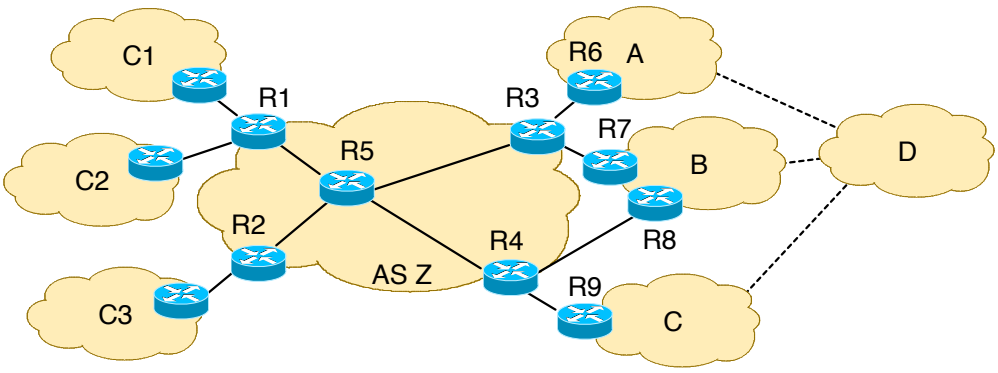


Figure 2.6: AS Z has multiple interdomain routes for destination D

$R3$) independently. To avoid routers in neighboring domains having to decapsulate packets, egress routers need to remove the encapsulation header before sending the packets to the next-hop router, using technique similar to the penultimate hop popping [26]. Similar to transit traffic originated from other ASes, traffic originated within the AS itself can also be forwarded to the correct egress links using tunneling.

2.5.2 Route Dissemination Within an AS

A prerequisite for an edge router to provide meaningful "customized" route-selection services is that it needs to have multiple available routes to choose from (otherwise, all its neighbors would inevitably receive the same route). Unfortunately, the way BGP routes are disseminated within today's ASes makes such "route visibility" often impossible. For example, in Figure 2.6, the AS Z as a whole learns four routes to D from four different neighboring edge routers ($R6$, $R7$, $R8$, $R9$). However, as BGP only allows a router to select and announce a single route for a destination, router $R5$ will only learn two of the available routes, one from $R3$ and $R4$. Even worse, $R1$ and $R2$ will only learn the one route selected by $R5$. For similar reasons, in large ASes where route reflectors are commonly used for better scalability, most edge routers have significantly reduced visibility of BGP routes [94].

Two different approaches can be used to provide better route visibility to the edge routers of

an AS—a distributed approach and a (logically) centralized one. In the distributed approach, a router in the AS needs to be able to *disseminate* multiple routes (per destination) to each neighbor. For backwards compatibility, this can be achieved by using multiple internal BGP (iBGP) sessions between routers. The BGP ADD-PATH extension, which supports the dissemination of multiple routes (per destination) through one BGP session [56], makes the dissemination process much more efficient. It is worth noting that, depending on how much flexibility an AS plans to provide, *not* all available routes need to be disseminated. For example, if an AS decides to have a couple of notions of "best routes" (e.g., best of all routes, and best of customer-learned routes), it only needs to disseminate at most two routes per destination (one of which must be a customer-learned route). Different ASes can make different trade-offs between the overhead of disseminating more routes within their own networks and the benefit of providing more routes to their neighbors to choose from.

Alternatively, an AS can also improve its route visibility by using a logically-centralized Routing Control Platform (RCP) [15, 97, 103]. In this case, an AS can deploy a set of servers in its network, each of which has a complete view of all available BGP routes. These servers then select routes on behalf of all the edge routers and install the selected routes to the respective routers. This logically-centralized approach can provide complete route visibility to the route-selection process with good scalability and performance [15, 97, 103]. As the desire for more flexible route selection grows, an RCP-like approach starts to make more sense, as it imposes less burden on route dissemination within an AS than the distributed approach.

2.5.3 Control Over Customized Selection

A big motivation of NS-BGP is to enable individual ASes to provide customized route-selection services to their neighbors. Therefore, an NS-BGP-enabled AS needs to take its neighbors' preferences of routes into account when selecting routes. Here we describe how an AS can control the amount of customer influence over its route-selection process, and how the customized route

selection can be realized.

An AS i can use different models to grant its neighbor j different levels of control over the ranking function λ_j^i. For example, AS i could adopt a *"subscription"* model, in which it offers several different services (ranking functions) for its neighbors to choose from, such as "shortest path", "most secure", and "least expensive". A neighbor j has the flexibility to decide which one to "subscribe" to, but does not have direct influence on how the ranking functions are specified. Although more flexible than conventional BGP, this model is a still fairly restrictive. For neighbors that require maximum flexibility in choosing their routes, an AS could offer a *"total-control"* model. In this model, AS i gives neighbor j direct and complete control over the ranking function λ_j^i. Therefore, j is guaranteed to receive its most preferred routes among all of i's available routes. For neighbors that require a level of flexibility that is in between what the previous two models offer, an AS could adopt a third, *"hybrid"* model. In this model, neighbor j is allowed to specify certain preference to AS i directly (e.g., avoid paths containing an untrusted AS if possible). When determining the ranking function λ_j^i for j, i takes both j's preference and its own preference into account (as the "best route" according to j's preference may not be the best for i's economic interest). Nevertheless, i still controls how much influence ("weight") j's preference has on the ranking function λ_j^i.

In Chapter 3, we describe in detail how these different models can be implemented by using a new, weighted-sum-based route-selection process with an intuitive configuration interface [103]. When deciding which model(s) to offer, an AS needs to consider the *flexibility* required by its neighbors as well as the *scalability* of its network, as the three service models impose different resource requirements on the provider's network. For example, the "subscription" model introduces the least overhead in terms of forwarding table size, route dissemination and customized route selection (e.g., each edge router or RCP server only needs to run a small number of route selection processes). On the other hand, the "total-control" model, while providing the finest grain of customization, imposes the most demanding requirements on system resources and results in the

highest cost for the provider. Therefore, we expect an AS to only provide such service to a small number of neighbors for a relatively high price. Since the costs of offering the three types of service models are in line with the degrees of flexibility they offer, we believe that an AS can economically benefit from offering any one or more of these models with appropriate pricing strategy.

It is worth mentioning that the "hybrid" and "total-control" models can be realized in two different ways. The simpler way is that the neighbor j tells the AS i what λ_j^i to use, so i only needs to select and export one route to j. The other way is that i announces all exportable routes to j, and j selects amongst them itself. The latter approach allows the j to hide its policy (ranking function) but requires i's ability to export multiple routes to j, and j's ability to directly tunnel its traffic to i's egress links. Finally, the NS-BGP safety conditions (Theorem 2.3.3) hold regardless of which one(s) of these three models are used.

2.6 NS-BGP and Incentives

Most studies of BGP make the implicit assumption that ASes will obediently adhere to the protocol. Recently, there has been a surge of interest in BGP's *incentive-compatibility* properties [36, 37, 67, 86, 44], motivated by the fact that ASes are independent entities with different, sometimes competing, economic interests. Given that NS-BGP provides a new interdomain route selection model, we are curious about its incentive-compatibility properties, and how these properties compare to BGP's. In this section, we examine NS-BGP from a game-theoretic perspective, and explore the possibility of making it incentive compatible. Unfortunately, we find that, as in conventional BGP, rational ASes have an incentive to lie about the paths they are using in NS-BGP. Therefore, unlike the positive routing stability results presented earlier in this chapter, the transition from BGP to NS-BGP does *not* improve the incentive-compatibility properties of a routing system. However, we argue that NS-BGP (and BGP) will remain stable even in the presence of protocol manipulation.

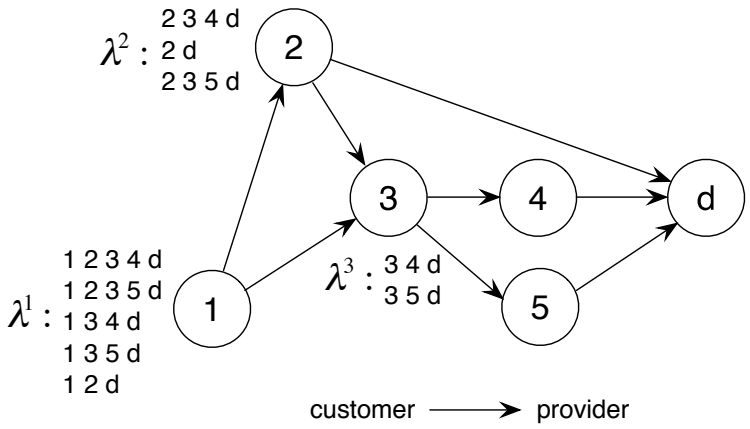


Figure 2.7: A system that is not incentive compatible in both BGP and NS-BGP

2.6.1 Background: BGP is Not Incentive-Compatible

Informally, saying that "BGP is incentive compatible" means that if all other ASes are following the rules of BGP, then the best course of action for an AS is to do the same (i.e., it has no incentive not to do so). We refer readers to [67] for an explanation of this framework.

Unfortunately, as observed in [67], conventional BGP is not necessarily incentive compatible even in small networks. This problem is further aggravated in realistic commercial settings in which ASes might be interested in attracting traffic from customers [44] to make more profit. Here we first illustrate the incentive-related problems with BGP using a simple example. This example also helps us in our later analysis of the incentive-compatibility properties of NS-BGP.

Consider the simple example illustrated in Figure 2.7, in which all three "Gao-Rexford" safety conditions hold. Assume that for AS 3, its main interest is attracting AS 1's traffic (i.e., making AS 1 forward traffic *directly* to AS 3), which is more important than attracting 2's traffic, which, in turn, is more important than the path it uses to send its outgoing traffic to d. Further, assume that 3 is bound by business contracts to provide connectivity to its customers, and thus must always announce *some* path to ASes 1 and 2. Observe that if AS 3 announces the path $(3\ 4\ d)$ (its most preferred route for outgoing traffic) to AS 2, AS 2 will choose path $(2\ 3\ 4\ d)$ and let AS 1 get its most preferred path $(1\ 2\ 3\ 4\ d)$. However, if AS 3, even though still only using path $(3\ 4\ d)$ to

forward all the traffic to d, announces the path $(3\ 5\ d)$ to AS 2 (but still announces path $(3\ 4\ d)$ to AS 1), AS 2 will choose the path $(2\ d)$ and announce it to AS 1. This way, AS 1 will choose path $(1\ 3\ 4\ d)$ and forward its traffic directly through AS 3. This example shows that AS 3 can improve its gain by announcing a path that it is *not* actually using to one of its customers. This inconsistency between the path AS 3 announces AS 2 and actual path it uses to forward 2's traffic is clearly an anomaly that should not happen (and is not expected by AS 2).

2.6.2 NS-BGP is Not Incentive-Compatible

We observe that the above counter-example for the incentive-compatibility of BGP can be easily extended to the NS-BGP setting. Now, assume that ASes 1 and 2 made their ranking functions known to their provider AS 3. If AS 3 honestly follows NS-BGP, it should announce path $(3\ 4\ d)$ to AS 2 (as it knows path $(2\ 3\ 4\ d)$ is AS 2's most preferred path). However, as in the BGP case, doing that will cause AS 3 to lose AS 1's (direct) traffic. If AS 3 ignores AS 2's ranking function, and announces path $(3\ 5\ d)$ to AS 2 instead, it will be able to attract AS 1's (direct) traffic and improve its gain. This simple example shows that ASes may have incentive to deviate from NS-BGP even in routing systems where the NS-BGP safety conditions hold.

2.6.3 Not Being Incentive-Compatible Does Not Affect Stability

We argue that BGP and NS-BGP not being incentive compatible in general does *not* necessarily mean that the respective routing systems will become unstable in the presence of unorthodox protocol manipulations. That is, while ASes might improve certain kinds of individual gains by manipulating these protocols, such actions are unlikely to affect the global routing stability.

This is because both the BGP safety conditions (the "Gao-Rexford" conditions) and the NS-BGP safety conditions (Theorem 2.3.3) are *motivated by* and *descriptive of* the actual economic interests of ASes, and therefore *reflect* ASes' behaviors in reality. Hence, an AS does not have an

economic incentive to violate the *export condition* (and carry transit traffic from peers or providers for free), or the *topology condition* (and serve as its own direct or indirect "provider"). Given these observations, we argue that, while ASes can manipulate NS-BGP in various ways, they have no incentive (and are unlikely) to break the NS-BGP safety conditions that guarantee global routing stability. Nevertheless, the lack of incentive compatibility of BGP and NS-BGP can cause problems like inconsistencies between the path announced by an AS and the actual path it uses to forward traffic. Hence, identifying sufficient conditions for incentive compatibility remains an important research problem.

2.7 Related Work

This chapter has two main areas of related work: more flexible interdomain route selection and interdomain routing stability. Recently, there has been an increase in the interest of providing more flexibility in interdomain route selection, from theoretical formalism and modeling of policy-based routing with non-strict preferences [20], to stability conditions of interdomain route selection for traffic engineering [111], to Routing Control Platform (RCP)-type systems that provide various degrees of customization support in BGP route selection [96, 97, 103].

A huge amount of effort has been put into understanding BGP's stability properties. Griffin *et al.*'s seminal work modeled BGP as a distributed algorithm for solving the *Stable Paths Problem*, and derived a theoretic sufficient condition (i.e., "No Dispute Wheel") for BGP stability [48]. Gao *et al.* proved a set of three practical conditions (i.e., the "Gao-Rexford" conditions) that guarantees BGP stability and also reflects the common business practices in today's Internet [43]. Gao *et al.* later extended their results to cover backup routing with BGP protocol extension and preference guidelines [42]. Feamster *et al.* explored the trade-off between the expressiveness of rankings and interdomain routing safety, and found if ASes are granted with complete flexibility with export (filtering) policies (i.e., can violate the export condition of the "Gao-Rexford" conditions), only

shortest-paths based ranking can guarantee stability [35].

2.8 Summary

This chapter presents Neighbor-Specific BGP (NS-BGP), an extension to BGP that provides both great *practical* benefits to ASes that deploy it and new *theoretical* contributions to the understanding of the fundamental trade-off between local policy flexibility and global routing stability. The NS-BGP model we propose enables individual ASes to offer customized route-selection services to neighbors. We prove that, comparing to conventional BGP, a less restrictive sufficient condition can guarantee the stability of the more flexible NS-BGP. Our stability conditions allow an AS to select *any* exportable routes for its neighbors without compromising global stability. We also show that NS-BGP remains stable even in partial deployment and in the presence of network failures, as long as the stability conditions are followed. We discuss the practical issues associated with deploying NS-BGP and show it can be readily deployed by individual ASes independently.

Chapter 3

Morpheus: Making Flexible Policies Easier to Configure

3.1 Introduction

In Chapter 2, we made the case that large ISPs have the incentive and the path diversity to meet the diverse requirements of its customers. We also proved that Neighbor-Specific BGP is a safe model that an ISP could use to realize routing policies that make flexible *trade-offs* amongst many different objectives, such as implementing business relationships with neighboring domains, providing good end-to-end performance to customers, improving the scalability of routing protocols, and protecting the network from attacks [16].

However, the *configurability* of ISP networks, i.e., the degree to which networks can be *customized* to implement flexible routing policies, is limited because of the unnatural restrictions that BGP imposes on the way ISPs select routes. BGP was designed when the Internet consisted of a small number of autonomous systems (ASes). Given the very limited path diversity within the small set of ASes, there was little need for a route selection process that supports configuration of flexible routing policies. However, as the Internet started to grow and path diversity increased,

network operators started to demand more flexibility to configure more complex policies. The response from the vendors and standards communities was an incremental "patchwork" of backward compatible features to add attributes and steps to the BGP decision process [1]. (For example, AS_PATH was introduced in BGP-2, NEXT_HOP was introduced in BGP-3, and LOCAL_PREF was introduced in BGP-4.) The outcome was a decision process that is counter-intuitive and notoriously hard to configure. Today, despite the rich path diversity available to large ISPs, configurability is limited by restrictions imposed by virtually every aspect of policy configuration such as the routing architecture, the BGP software implementation, and its configuration interface.

For instance, each BGP router selects a single "best" route for each prefix, forcing all neighboring ASes connected to the same edge router to learn the same route, even if some customers would be willing to pay a higher price to use other routes. Within each router, the standard BGP implementation selects routes only based on the attributes of the BGP updates, falling short of realizing routing policies that, for example, require using outside measurement data. Finally, current BGP decision process imposes inherent restrictions on the policies an ISP can realize [80]. Consisting of a series of tie-breaking steps, the BGP decision process compares one attribute at a time until only one best route remains. The ordering of steps imposes a strict *ranking* on the route attributes, making it impossible to realize flexible policies that make *trade-offs* between policy objectives. For example, a useful policy that strikes a balance between revenue and route stability could be: *"If all routes are unstable, pick the most stable path (of any length through any kind of neighbor), otherwise pick the shortest stable path through a customer (then peer, and finally provider)."* However, this seemingly simple policy cannot be realized today. In addition, policy objectives that are not part of the original BGP protocol, such as security and performance, are hard to add into its decision process, even if the importance of these objectives becomes obvious over time.

Stepping back, we ask the question: "Starting from a clean slate, how can we *design for configurability*?" That is, instead of seeking the best way to configure the existing system, we design a new system with configurability as a first-order goal. To make the new system practically adopt-

able, we focus on solutions that do not require cooperation between domains. Since ISPs are often business competitors, cooperation among them has proved notoriously difficult in practice. This constraint essentially prevents changes to the *interdomain* routing protocol that require collaboration of multiple domains. Fortunately, such changes are not necessary—as mentioned in Section 2.1, large ISPs have a lot of path diversity, and can safely and effectively "act alone" in applying many flexible routing policies.

To design for configurability, we consider the following **route selection problem** an ISP faces: *Given a set of available routes $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ for a prefix p, choose a best route r^* for each router according to a set of criteria $\mathcal{C} = \{c_1, c_2, ..., c_k\}$.* The set of criteria (i.e., policy objectives) includes route characteristics such as stability, security, and performance. These criteria may be conflicting in the sense that no route is the best with respect to all criteria simultaneously. Therefore, to design for configurability, the routing system must ensure that the network administrator has the flexibility to make arbitrary trade-offs among the criteria. Our solution to the route selection problem is a system that we call *Morpheus* as it gives ISPs the power to "shape" their routing policies. Morpheus relies on the following system components:

• A **routing architecture** that is responsible for (1) learning the "inputs" and (2) disseminating the "outputs" of the route selection problem. The routing architecture allows a set of *Morpheus servers* to choose the best routes from the set $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ of *all* routes available to the AS, and ensures that the servers can assign any route in $\mathcal{R}$ *independently* to each neighbor without restrictions.

• A **server software architecture** giving the network operators the ability to make trade-offs among the criteria $\{c_1, c_2, ..., c_k\}$. It includes a set of *policy classifiers* and one or more *decision processes*. Each classifier tags routes with criteria-specific labels. The decision process computes a cumulative score as a weighted sum of the labels for each route and picks the route with the highest score. To pick potentially different routes for different neighbor networks (as supported

by the routing architecture), multiple decision processes (possibly one per neighbor) can run in parallel.

• A **configuration interface** through which network operators can configure the decision processes. The straightforward method for a network operator to configure a decision process is to directly specify a weight for each criterion. However, without a systematic procedure for determining what the weights should be, this method would be error prone. Morpheus provides such a systematic procedure based on the Analytic Hierarchy Process (AHP) [85], which derives the appropriate weights based on operator's preferences on the policy objectives.

We have implemented Morpheus as a routing control platform consisting of a small number of servers that select BGP routes in a logically centralized way. Previous work on centralized routing platforms [15, 97, 96] has demonstrated that they can be made scalable and reliable enough for deployment in large ISP networks without sacrificing backwards compatibility. However, the previous work mainly focused on the *feasibility* of such logically centralized system, stopping short of addressing the poor configurability of BGP policies. In particular, the previous work did not identify the necessary supports for configurability from the routing architecture and proposed only limited improvements in the BGP decision process and its configuration interface.

The rest of the chapter is organized as follows. In Section 3.2, we identify the necessary changes to the current routing architecture in order to support flexible policies. We present the software architecture of the Morpheus server in Section 3.3, and give examples on how to configure routing policies through its AHP-based configuration interface in Section 3.4. Section 3.5 describes the prototype implementation of Morpheus as extension to the XORP software router [53]. Section 3.6 presents the evaluation of the prototype Morpheus server and demonstrates that the gain in flexible policies does not come at the expense of scalability and efficiency. Finally, we present related work in Section 3.7 and summarize the chapter in Section 3.8.

3.2 Routing Architecture

In this section, we present the intra-AS routing architecture of Morpheus, which enables us to replace the BGP decision process with a new, flexible route selection process in Morpheus servers (Section 3.3). We propose three changes to the way routes are disseminated and assigned, and the way traffic is forwarded within an AS, which provides the ultimate flexibility to the "inputs" and "outputs" of the route selection problem formulated in the Introduction. These changes enable Morpheus to: (1) have complete visibility of all alternative routes, (2) assign customized routes to different edge routers in the AS and neighboring domains, and (3) assign routes independently of each other without causing forwarding loops. As a result, the route selection process can assign any available route to any ingress link (i.e., neighbor) independently. All three architectural features are incrementally deployable through configuration changes and do not require hardware or software upgrades to existing routers.

3.2.1 Complete Visibility of BGP Routes

As discussed in Section 2.1, path diversity is the basis of policy flexibility. However, much of the path diversity of a large ISP remains unused as routers do not have complete visibility of BGP routes [94]. An edge router may learn multiple routes for the same destination prefix through external BGP (eBGP) sessions with neighbor ASes. However, the router can only select and propagate one best route per prefix to other routers in the AS. As a result, there are many routes visible to only one router in an AS. For example, in Figure 2.6, R3 and R4 each learns two routes to destination D, but can only propagate one to R5 (say, the one via R6 and R8, respectively). R5, in turn, propagates only one route (say, the one via R8) to R1 and R2. Then, R2 does not learn, and hence cannot use any of the other available routes (via R6, R7, or R9), even if it would have been preferred by the customer C3 (e.g., to avoid having its traffic go through AS B). Such loss of visibility gets even more pronounced in large networks due to the use of route reflectors [94]. Although propagating

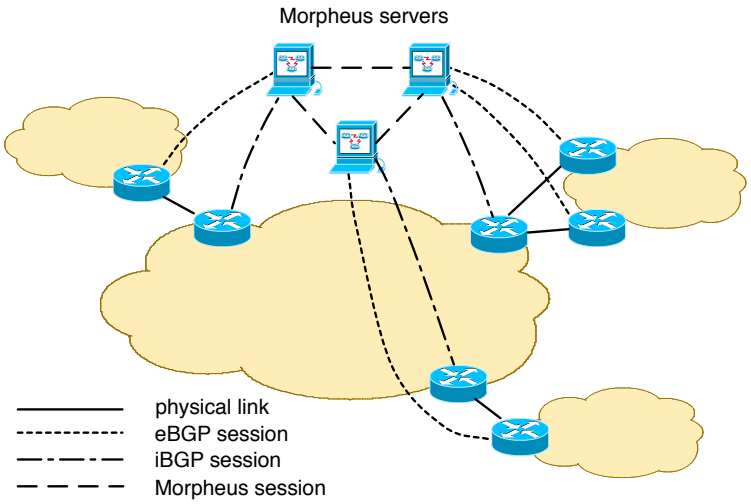


Figure 3.1: Morpheus routing architecture: Morpheus servers peer with neighboring domains via multi-hop BGP sessions; edge routers direct interdomain traffic through tunnels.

only one route helps limit control-plane overhead, it imposes significant limitations on flexibility.

Design Decision 1: *An AS should have complete visibility of eBGP-learned routes to enable flexible routing policies.*

Morpheus uses a small collection of servers to select BGP routes on behalf of all the routers in the AS, as shown in Figure 3.1. Morpheus can obtain full visibility of all available BGP routes through (multi-hop) eBGP sessions with the routers in neighboring ASes, as in the Routing Control Platform [33, 97].[1] Morpheus assigns BGP routes using internal BGP (iBGP) sessions between the servers and the routers for backwards compatibility. The Morpheus servers also ensure that the BGP routes propagated to eBGP neighbors are *consistent* with the routes assigned to the associated edge links. For example, in Figure 2.6, if Morpheus assigns C3 the route through R6 to reach D, it must also propagate the same route to R2 (the edge router C3 is connected to), so that R2 knows how to forward C3's traffic to D using the expected path. Since this architecture uses the BGP protocol itself to learn and assign routes, it does not require any upgrade to the routers in the ISP.

[1] Alternatively, full visibility of the routes can be obtained through BGP Monitoring Protocol (BMP) sessions [88] with the AS's own edge routers, which is more scalable. The ways in which workload is divided among Morpheus servers and consistency is maintained among them are similar to those of previous RCP system [15, 97].

3.2.2 Flexible Route Assignment

Even *with* complete visibility of alternative routes, today's BGP-speaking routers cannot assign different paths to different customers. In Figure 2.6, the two customers C1 and C2 connected to the same edge router R1 may want to use the two different paths through the same egress point R3 to reach D, respectively. To make such policy possible, the AS must have the ability to (1) use available paths through any *egress link* (rather than *egress router*) flexibly, and (2) assign those routes to the ingress links *independently* (whether or not they connect to the same edge router).

Design Decision 2: *An AS should be able to assign any route through any egress link to any ingress link independently.*

With full visibility of all eBGP-learned routes, Morpheus can easily pick the best routes through any egress link for its customers and edge routers individually. Morpheus can disseminate multiple routes per prefix to edge routers in several ways.[2] Since the edge routers are no longer responsible for propagating BGP routing information to neighbor ASes, Morpheus does not need to send all of the route attributes—only the destination prefix and next-hop address are strictly necessary. This enables a significant memory reduction on edge routers. Upon receiving these routes, edge routers can use the "virtual routing and forwarding (VRF)" feature commonly used for MPLS-VPNs to install different forwarding-table entries for different customers [77].

3.2.3 Consistent Packet Forwarding

With the flexibility of assigning any route through any egress link to any neighbor independently, extra care needs be taken in the data plane to avoid introducing forwarding loops. When a router has multiple "equally good" routes, it is common practice to pick the route through the "closest" egress point, based on the Interior Gateway Protocol (IGP) weights, a.k.a. hot-potato routing. For

[2]This can be achieved by using the "route target" attributes commonly used with VRF in MPLS-VPN [77], or having multiple iBGP sessions between a Morpheus server and an edge router. Other options include using the BGP "add-paths" capability [100].

example, in Figure 2.6, if the routes to D through link R3-R6 and link R4-R9 have the same local preference and AS-path length, and if R1 is closer to R3 than to R4 (in terms of IGP weights), R1 will pick the route through R3-R6. Hot-potato routing ensures consistent forwarding decisions among the routers in the network. For example, if R1 picks the route through R3-R6 to reach D, other routers on the forwarding path (i.e., R5 and R3) are guaranteed to make the same decision.

However, hot-potato routing introduces problems of its own. First, it significantly restricts the policies an AS can realize. For example, in Figure 2.6, R1 and R2 connect to a common intermediate router R5. Hot-potato routing forces them to use the same egress point, rather than allowing (say) R1 to use R3 and R2 to use R4. In addition, a small IGP change can trigger routers to change egress points for many prefixes at once, leading to large traffic shifts and heavy processing demands on the routers [93].

Design Decision 3: *The routers in an AS should forward packets from the ingress link to its assigned egress link.*

To achieve this goal, Morpheus relies on IP-in-IP or MPLS tunnels to direct traffic between edge links. This design choice offers several important advantages, beyond allowing flexible route assignment without the risk of forwarding anomalies. First, Morpheus can rely on the IGP to determine how traffic flows between ingress and egress routers, reducing the complexity of the Morpheus server and ensuring fast reaction to internal topology changes. Second, Morpheus does not need to select BGP routes for the internal routers, reducing the total number of routers it has to manage. MPLS or IP-in-IP tunneling is readily available at line rate in many commercial routers, and a "BGP-free core" is increasingly common in large ISPs. In Morpheus, packets are tunneled between edge *links* (rather than between edge routers as is common today). To avoid routers in neighboring domains (e.g., R6 in Figure 2.6) having to decapsulate packets, edge routers (e.g., R3) need to remove the encapsulation header as part of forwarding the packets, using technique similar to penultimate hop popping [26].

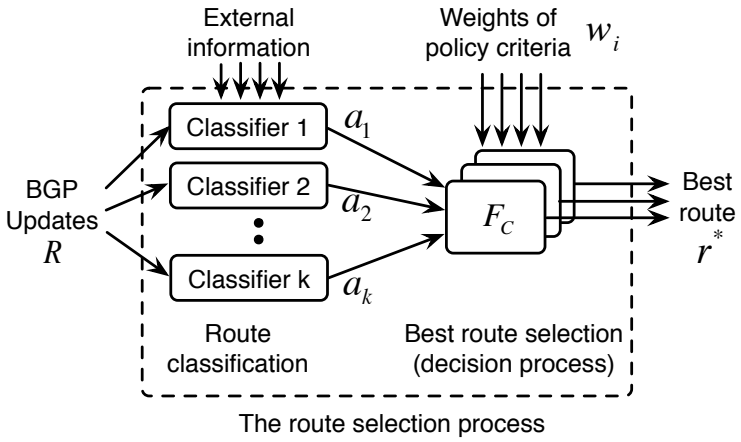


Figure 3.2: Morpheus' BGP route selection process, which includes route classification and best route selection.

3.3 Server Software Architecture

The Morpheus server needs to solve the **route selection problem** introduced in Section 3.1: *Given a set of available routes $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ for a prefix p, choose a best route r^* according to a set of criteria $\mathcal{C} = \{c_1, c_2, ..., c_k\}$ for each neighboring router.* This problem naturally devolves into two main steps: (i) *classifying* the routes based on each criterion and (ii) *selecting* the best route based on the set of criteria, as shown in Figure 3.2. Each *policy classifier* tags every received route based on a single policy objective. Each *decision process* picks a best route according to the tags using a "decision function" $\mathcal{F}_\mathcal{C}$ that is configured to realize a particular routing policy. A Morpheus server can run multiple decision processes in parallel, each with a different routing policy, to pick customized routes for different neighbors.

3.3.1 Multiple Independent Policy Classifiers

The introduction of policy classifiers provides flexibility by providing a separate attribute for each policy objective, and incorporating "side information" into route selection.

Separate Attribute for Each Policy Objective

The BGP decision process selects best routes by examining one BGP attribute at a time, e.g., first "local-preference", followed by "AS-path length" and so on. As BGP policies involve more and more policy objectives, many of them are forced to be realized by using the same BGP attribute. For example, to realize the common business relationship policy of "prefer customer routes over peer routes, and prefer peer routes over provider routes", customer / peer / provider routes could be assigned with local-preference value of 100 / 90 / 80, respectively. At the same time, operators often increase or decrease the local-preference of a route to make it more or less favorable in the decision process to control the traffic load of certain links. In fact, many other complicated rules are also overloaded to "local preference" via mechanisms such as "route-maps" to *indirectly* influence BGP's multi-stage decision process. The lack of separate attributes for individual policy objectives causes policy configuration to become immensely convoluted, as the attribute overload becomes more severe.

Design Decision 4: *A Morpheus server should use a separate attribute for each policy objective.*

Morpheus' policy classifiers realize this design decision by *tagging the routes*. Each classifier takes a route as input, examines the route according to a specific policy criterion, and generates a tag that is affixed to the route as metadata. For example, a business-relationship classifier may tag a route as "customer", "peer", or "provider"; a latency classifier may tag a route with the measured latency of its forwarding path; a loss classifier may tag a route with the measured loss rate of the path; a stability classifier may tag a route with a penalty score that denotes the instability of the route (using, for example, a route-flap damping algorithm [98]); a security classifier that detects suspicious routes (e.g., those being hijacked) may tag a route as "suspicious" or "unsuspicious" [59].

Each policy classifier works independently and has its own tag space, obviating the need to overload the same attribute. It also makes it easy to extend the system with a new policy objective

by adding a new classifier, without changing or affecting any existing ones. Furthermore, when a new module needs to be incorporated into the system, upgrades need only be applied to the Morpheus servers instead of all routers in the AS. These classifier-generated tags are purely local to Morpheus, and are never exported with BGP update messages; as such, using these tags does not require any changes to any routers.

By tagging the routes, rather than filtering or suppressing them, the decision process is guaranteed to have full visibility of all valid candidate routes (except those that are ill-formed or cannot be used under any circumstances, e.g., those with loops in their AS paths). This is in sharp contrast to the current BGP implementation in which all the routes for the same prefix may be filtered or suppressed (e.g., in the case of route-flap damping), sometimes leaving the decision process with no route to choose from.

Incorporate Side Information

Another issue that limits the flexibility of routing policies is the lack of *side information*. Many useful routing policies require additional information that is not part of the BGP updates. For example, to select the route with the shortest latency to a destination, we need performance measurement data. (As mentioned in Section 2.1.1, the AS-path length is a poor indicator of path latency.) In general, side information about route properties includes *external information* such the business relationships with the neighbors, measurement data, or a registry of prefix ownership, and *internal states* such as a history of ASes that originated a prefix (which can be used to detect prefix hijacking [59]), or statistics of route instability. However, there was no systematic mechanism to incorporate side information in routers. Network operators had to either "hack" their BGP configurations in an indirect and clumsy way (e.g., tweaking "route-maps"), or wait for software upgrades from router vendors (if the need for certain side information becomes compelling) and then upgrade a large number of routers [3].

[3]Recently, several RCP-type systems started to offer the similar ability to incorporate side information [96, 97].

Design Decision 5: *A Morpheus server should be able to use external information and / or keep internal state when determining the properties of routes.*

The introduction of policy classifiers makes it easy to incorporate side information as each policy classifier can have access to different external data sources containing the information needed to classify the routes. For example, the business-relationships classifier can have access to up-to-date information about the ISP's business relationships with neighboring ASes through a corresponding configuration file. A latency classifier and a loss classifier can get measurement information about path quality from a separate performance monitoring system, or a reputation system (e.g., AS X is well known to have long latency or a high loss rate). A security classifier can have access to a registry of prefixes and their corresponding owners.

Different classifiers can also maintain separate internal states. For instance, a stability classifier can maintain statistics about route announcement and withdrawal frequencies. A route security module that implements Pretty Good BGP (PGBGP)—a simple algorithm that can effectively detect BGP prefix and subprefix hijacks—can keep past history of BGP updates in the past h days (where h is a configurable parameter) [59].

Care needs to be taken when taking performance metrics (e.g., latency and loss) into the decision process, as these properties of a path could potentially change quickly with time. Recent studies suggest that it is possible to factor performance into route selection in a stable way [58, 54]. We plan to further investigate the trade-off between route stability and the responsiveness of route selection to performance changes in the context of Morpheus (e.g., use a timer in the classifiers to control how often the performance properties of routes change in the decision process).

3.3.2 Multiple Weighted-Sum Decision Processes

The Morpheus server uses a weighted-sum decision process to realize trade-offs amongst different objectives. It also supports running multiple decision processes in parallel to realize different customized policies simultaneously.

Weighted-sum for Flexible Trade-offs

The conventional step-by-step BGP decision process imposes a strict ranking of route attributes, starting with local preference and followed by AS-path length and so on. As a result, policies that strike a trade-off among policy objectives are hard to realize, such as the example mentioned in Section 3.1 that balances stability and business relationships.

Design Decision 6: *The Morpheus decision process should support trade-offs among policy objectives.*

To achieve this goal, the decision function $\mathcal{F}_{\mathcal{C}}$ in the route selection problem formulation (as mentioned in Section 3.1) must allow trade-offs among policy objectives. A simple, yet powerful method is the *weighted-sum*. For example, for a route $r \in \mathcal{R}$ (where $\mathcal{R}$ is the set of alternative routes), its weighted-sum *score* is:

$$S(r) = \sum_{c_i \in \mathcal{C}} w_i \cdot a_i(r) \tag{3.1}$$

where w_i is the *weight* for criterion c_i in $\mathcal{C}$, and $a_i(r)$ is route r's *rating* of criterion i. For a prefix p, the decision function $\mathcal{F}_{\mathcal{C}}$ selects the route with the highest score as the best choice:

$$r^* = \mathcal{F}_{\mathcal{C}}(r) = \arg\max_{r \in \mathcal{R}(p)} S(r) \tag{3.2}$$

We choose the weighted sum as the basis of Morpheus' decision process for three reasons. First, the weighted sum provides an expressive way to make trade-offs between the criteria through the configuration of their weights, and it can also be used to express a sequential process like the standard BGP decision process. Second, weighted sums are simple to compute and thus well-suited to making routing decisions in real time. Third, it allows us to leverage Analytic Hierarchy Process (AHP), a technique in decision theory, to design a simple and intuitive configuration interface, which can automatically derive the weights according to operator's preferences on policy

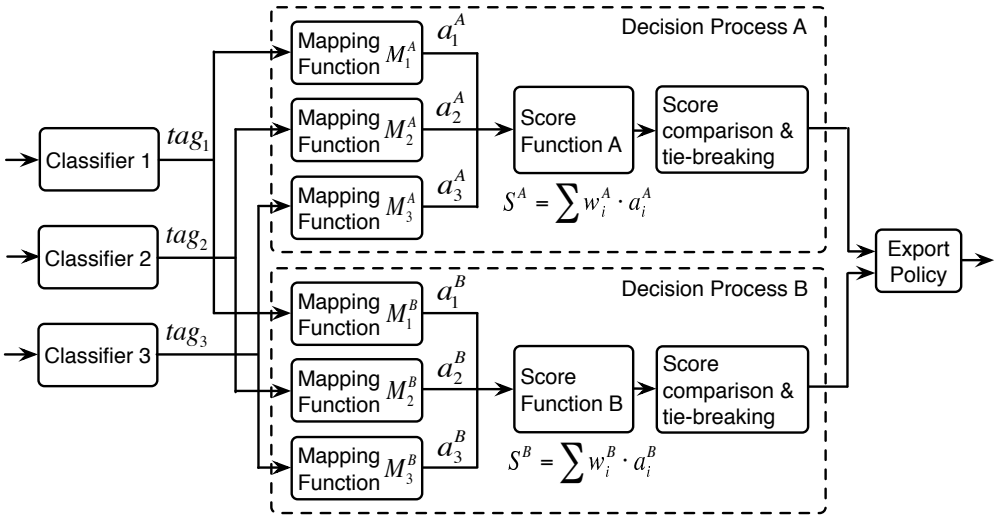


Figure 3.3: Each decision process consists of a set of mapping functions of the policy objectives and a score function. Different decision processes are configured with different mapping functions and/or score functions to realize different policies.

objectives (as discussed in Section 3.4).

Morpheus instantiates one decision process for each routing policy and supports running multiple decision processes in parallel. To allow different decision processes to *interpret* a policy tag differently, each decision process has a set of "mapping functions" before the "score function", as shown in Figure 3.3. The introduction of the mapping functions offers two major benefits.

First, the introduction of the mapping functions decouples the *generation* of tags (the job of the classifiers), and the *interpretation* of tags (the job of the mapping functions). This way, each policy classifier can tag routes in its own tag space without worrying about the consistency with other classifiers. This facilitates the implementation of classifiers by third parties. With the mapping functions, network operators can simply "plug and play" different classifier modules. The mapping functions can ensure that all tags are converted to the same uniform numerical space to make the comparison between different policy criteria meaningful. We believe this open platform will foster the sharing of classifier modules in the operations community and may also lead in the long run to the emergence of a market centered around these modules.

Second, the mapping functions enables different policies to interpret the same policy tag *dif-*

ferently. For example, one policy may want to set a threshold for route stability and treat all routes with penalty values below the threshold as "equally stable", while another policy may want to always select the most stable route available. As shown in Figure 3.3, the same tag tag_1 can be mapped to different ratings a_1^A and a_1^B by two different mapping functions $\mathcal{M}_1^A$ and $\mathcal{M}_1^B$. Therefore, network operators can realize different policies through different configurations of the mapping functions (as well as weights of the policy objectives), as illustrated by the examples in Section 3.4.

After passing the mapping functions, the route is sent to the score function which computes its score, as shown in Figure 3.3. Then the scores of all the routes for the same destination prefix are compared, and the route with the highest score is picked as the best route. If there are multiple routes with the same highest score, the operators have the choice to break the tie using different mechanisms, such as configuring a (potentially different) ranking of egress links for each ingress link, and pick the route with the highest egress link ranking as the best route [102]; or simply using router ID. As in conventional BGP, the export policy module after the decision process makes the final decision on whether to send the route out or filter it.

Parallel Decision Processes for Customized Policies

BGP allows an AS to influence how other ASes reach itself (e.g., through the use of BGP communities). However, BGP provides no mechanism for an AS to influence how its provider picks routes for it to reach the rest of the Internet. However, such coordination is increasingly important as more customers want routes with particular properties (e.g., low latency, high bandwidth, good security). For example, many content providers (e.g., social network Web sites) rely on their ISPs to reach their users (i.e., the "eyeballs"). To get closer to the "eyeballs", content providers commonly buy services from multiple transit providers and use only the routes that meet their performance requirements. This is not economical for the content provider. A transit provider that could flexibly assign the routes based on customers' preferences would have an advantage over

other ISPs in attracting customers.

Design Decision 7: *An AS should allow its neighbors (e.g., its customers) to influence its routing policies by specifying their preferences.*

To support different customer choices, Morpheus supports the realization of multiple independent routing policies simultaneously, through the parallel execution of multiple decision processes, each selecting its own best routes, as shown in Figure 3.3.

To avoid changing the BGP protocol, Morpheus uses an out-of-band communication channel for customers to specify preferences through a simple configuration interface. For example, the provider could allow a customer to independently and directly configure the weights in a decision process. Alternatively, the provider could combine the customers' preferences between certain policy objectives, and combine them with its own preferences through an AHP-based configuration interface (as discussed in Section 3.4). While providing a separate decision process for each customer may introduce scalability challenges, we believe in practice, the routes most customers want can be reduced to a handful of types, such as low-latency routes, most secure routes, most stable routes, low-cost routes. The provider could simply provide these options to its customers, and only provide customized decision processes to a very limited number of customers who demand more control of their routes.

In any case, Morpheus provides an AS the ability to select routes based on a variety of factors. However, this extra flexibility should not come at the expense of global routing instability. Fortunately, the NS-BGP stability conditions presented in Section 2.3 provide a useful guideline for ISPs that want to offer flexible policies on a per neighbor (or per neighbor group) basis. In addition, possible extensions of recent stable load-balancing techniques [31, 58, 39, 54, 7] can be explored to prevent oscillations in interdomain load-sensitive routing. In both cases, considerable flexibility in interdomain routing is possible without compromising global stability.

3.4 AHP-Based Policy Configurations

In this section, we present how to configure routing policies in Morpheus. In theory, operators could configure the mapping functions and the weights directly to realize policies. However, humans are not good at setting a large number of weights directly to reflect their preferences. Instead, studies show that humans do a much better job in expressing their preferences through pairwise comparisons between alternatives, even though the results of these comparisons are often inconsistent [85]. Based on this observation, Morpheus leverages the Analytic Hierarchy Process (AHP) [85], a technique in decision theory, to provide a simple, intuitive configuration interface. Network operators specify their policy preferences through pair-wise comparisons, and AHP automatically derives the weights of policy objectives and the appropriate ratings of the mapping functions. After briefly explaining how AHP works in an "offline" fashion, we propose an "online" version that is more appropriate for real-time route selection. We then show a policy configuration example, in which the ISP allows its customer to configure part of the decision process. At the same time, the ISP itself controls how much influence on the decision process the customer can have.

3.4.1 The Offline AHP Configuration Process

AHP is a well-studied, widely-applied technique in Multi-Criteria Decision Analysis [11], a field in decision theory. It provides a simple, yet systematic way to find the overall best choice from all alternatives, according to the decision maker's preferences of the alternatives with regard to individual criteria [85]. In interdomain routing policy, the alternatives are the available routes, the decision maker is the network operator, and the criteria are the policy objectives.

The first step in AHP is to model the decision problem as a *decision hierarchy*, as shown in Figure 3.4. At the bottom of the hierarchy are the *alternatives*, i.e., the possible solutions of the decision problem. One solution must be selected among the alternatives based on a set of *criteria*,

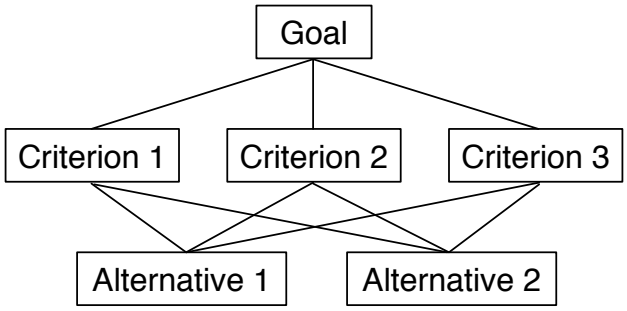


Figure 3.4: The decision hierarchy of AHP.

as shown in the middle of the hierarchy. For each criterion, the decision maker then performs pairwise comparisons of all alternatives. For each comparison, the decision maker specifies his / her preference of one alternative over the other using a number. The scale from 1 to 9 has proven to be the most appropriate [85], in which, when comparing criteria *p* to *q*, 1 means *p* and *q* are equally preferred, 3 means weak preference for *p* over *q*, 5 means strong preference, 7 means demonstrated (very strong) preference, 9 means extreme preference. The inverse values 1/3, 1/5, 1/7 and 1/9 are used in the reverse order of the comparison (*q* vs. *p*). Intermediate values (2, 4, 6, 8) may be used when compromise is in order.

Table 3.1: Comparison matrix

Loss Rate	$R1$ (0.01)	$R2$ (0.03)	$R3$ (0.05)	Weight
$R1$ (0.01)	1	3	9	0.69
$R2$ (0.03)	1/3	1	3	0.23
$R3$ (0.05)	1/9	1/3	1	0.08

An example is shown in Table 3.1, where three alternative routes $R1$, $R2$, and $R3$ are compared in pairs based on their loss rate. Note that although the table shows the entire matrix of 9 preferences, the operator only needs to specify 3 of them—"$R1$ vs. $R2$", "$R1$ vs. $R3$", and "$R2$ vs. $R3$". Here the operator weakly prefers $R1$ (with a loss rate of 0.01) over $R2$ (with a loss rate of 0.03); strongly prefers $R1$ over $R3$ (with a loss rate of 0.05); and weakly prefers $R2$ over $R3$. The table also shows the weights of all alternatives, which are computed from the principal eigenvector

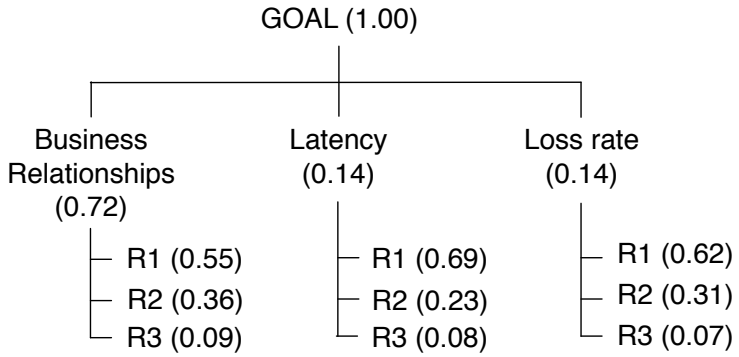


Figure 3.5: Example of a decision hierarchy.

of the preference matrix [85]. In this case, the operator's preferences are "consistent", i.e., "$R1$ vs. $R3$" (9) = "$R1$ vs. $R2$" (3) $\times$ "$R2$ vs. $R3$" (3), so the weights can be derived by normalizing the values in any column of the preference matrix. However, humans are likely to give *inconsistent* answers in a series of pair-wise comparisons, and AHP provides a systematic way to deal with inconsistency, as illustrated in the example in Section 3.4.3.

With operator's preference of alternative routes on each criterion (e.g., business relationships, latency and loss rate in Figure 3.5), AHP can derive the rating $a_i(r)$ of route r for each criterion i, as in Equation (3.1). To get the weight w_i of each criterion i, the operator also needs to determine the preference (relative importance) of different criteria through similar pair-wise comparisons of criteria. With the preferences of all criteria pairs, AHP can derive the appropriate weight for every criterion, and calculate the overall score of an alternative route using Equation (3.1). For example, in the hierarchy shown in Figure 3.5, $S(R1) = 0.72 \times 0.55 + 0.14 \times 0.69 + 0.14 \times 0.62 = 0.58$.

3.4.2 Adapting AHP to Work Online

Applying the conventional AHP technique to the route selection problem directly, as described in Section 3.4.1, only works in an *offline* fashion. This is because whenever a new route is received, a human operator has to compare all alternatives routes in pairs with regard to every policy objective (to get the rating $a_i(r)$), which can not be done in real time.

To make the AHP-based decision process work *online*, we replace the alternatives in the decision hierarchy with a set of *subcriteria*. For example, in Figure 3.6, the business relationships criterion can be divided into three subcriteria: customer, peer, and provider. This change allows network operators to specify their preferences on each set of subcriteria offline, while enabling the ratings $a_i(r)$ of received routes to be generated in real time. For example, for the business-relationship criterion, an operator can specify his / her preference of customer / peer / provider routes through pair-wise comparisons offline. The appropriate rating for each type of route will be derived by AHP automatically and stored in the mapping function (as shown in Figure 3.3).

In summary, the online, AHP-based policy configuration process can be performed in three steps: (1) **Decompose:** The network operator formulates the decision problem by identifying a hierarchy of criteria (and subcriteria); (2) **Specify preferences:** For each pair of criteria at the same level of the hierarchy and with the same "parent criterion", the network operator specifies his / her preference of one criterion over the other; (3) **Derive weights:** The preferences are organized in preference matrices and weights are derived by AHP using linear algebra operations [85]. Note that operators are only involved in the first two steps, and the third step is performed by the configuration program automatically.

3.4.3 A Policy Configuration Example

As mentioned in Section 3.3, Morpheus enables an ISP to get input from its customers about their preferences on routes. Here we give an example that shows how customer preference can be incorporated into the decision process using the AHP-based configuration interface.

Suppose the ISP has a customer C who is a content provider, and C has purchased the "premium service" that allows it to specify its preference on the routes it learns from the ISP. As a content provider, C is primarily interested in learning routes that have low latency to the destinations (i.e., to get the content closer to the "eyeballs"). The ISP, on the other hand, cares about the "business relationships" property of the routes, as it would earn profit by forwarding traffic through

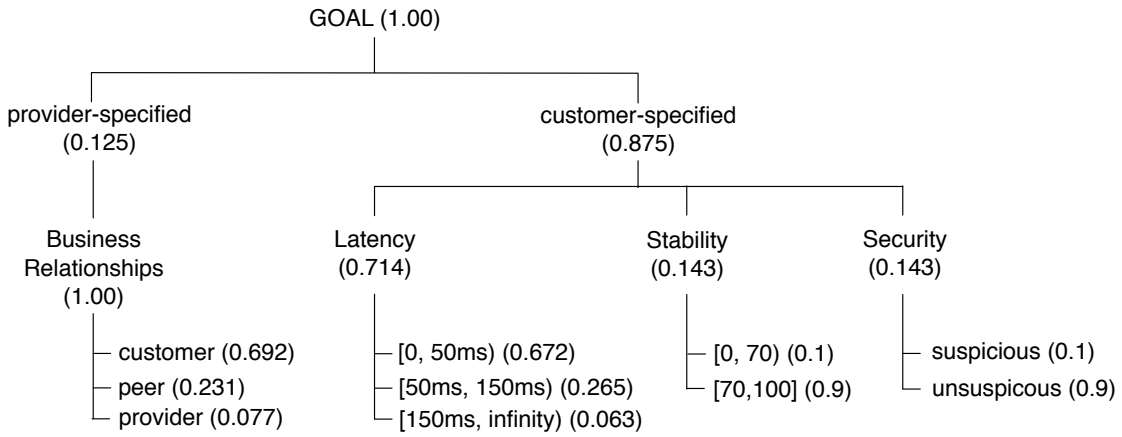


Figure 3.6: The AHP hierarchy of an example routing policy.

a customer, and it would have to pay to forward traffic through a provider.

Figure 3.6 shows the AHP hierarchy of the routing policy, which takes four policy objectives into account: business relationships, latency, stability, and security. As the first step of the configuration, the ISP needs to decide how much influence to the decision process it gives to customer C. As a premium service, the ISP allows C to directly specify its preferences on all policy objectives except business relationships. It also strongly prefers the customer-specified objectives over the provider-specified objective, and enters "7" in the "customer-specified vs. provider-specified" comparison. AHP then automatically derives the relative weights of the two types of objectives: 0.875 for the three customer-specified objectives (latency, stability, and security) and 0.125 for the provider-specified objective (business relationships).

To determine the relative weights of latency, stability, and security, the customer C needs to specify its preferences through pair-wise comparisons. Assuming that C enters "latency vs. stability" = 5, "performance vs. security" = 5, and "stability vs. security" = 1, AHP can then derive the weights of the three objectives: latency (0.714), stability (0.143), and security (0.143), as shown in Figure 3.6.

Now that the weights of the four policy objectives are derived, the ISP and the customer C only need to configure the corresponding mapping functions for the objectives. Assuming that the ISP

specifies its preferences on business relationships as: "customer vs. peer" = 3, "peer vs. provider" = 3, and "customer vs. provider" = 9, then AHP automatically derives the ratings of the three types of routes for the mapping function of business relationships. Upon receiving a route tagged as "customer", "peer", or "provider" by the business relationship classifier, the mapping function will assign it with a business relationship rating of 0.692, 0.231, or 0.077, respectively.

For the latency mapping function, suppose the customer C is given three latency intervals: i_1 = $[0, 50msec]$, i_2 = $[50msec, 150msec]$, and i_3 = $[150msec, \infty]$, and it has the following preferences: "i_1 vs. i_2" = 5, "i_1 vs. i_3" = 9, and "i_2 vs. i_3" = 3. AHP will then derive the ratings the mapping function should use to map the routes that fall into the three intervals: i_1 = 0.672, i_2 = 0.265, and i_3 = 0.063. While calculating the ratings, AHP also calculates the *consistency ratio* of the preferences [85], where a consistency ratio of 0 means all preferences are consistent. In this case, the three preferences are inconsistent (i.e., "i_1 vs. i_3" (9) $\neq$ "i_1 vs. i_2" (5) $\times$ "i_2 vs. i_3" (3)), and the consistency ratio is 0.028. AHP requires the consistency ratio to be no larger than 0.05 (n = 3), 0.08 (n = 4), or 0.1 ($n \geq 5$) for a set of preferences to be acceptable, where n is the number of alternatives [85]. (As 0.028 is below the 0.05 threshold, this set of preferences is acceptable.) When a set of preferences specified by an operator has a consistency ratio larger than the threshold, Morpheus will request the operator to reset the preferences.

For stability, we assume the stability classifier runs an algorithm similar to the one used by route-flap damping (RFD), and tags each route with a number between 0 and 100. The higher the number is, the more stable the route is. The customer C treats routes with a stability tag below 70 as unstable, and it extremely prefers stable routes over unstable ones. For security, we assume the security classifier runs the Pretty-Good BGP (PG-BGP) [59] algorithm, and tags every route as either "suspicious" or "unsuspicious". The customer C extremely prefers unsuspicious routes over suspicious routes.

In a similar fashion, the provider can provide customized routing policies to different customers using separate decision processes (as shown in Figure 3.3), and allow each customer to configure

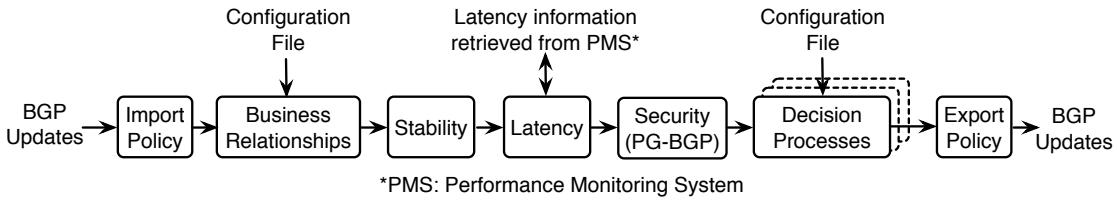


Figure 3.7: Morpheus prototype implemented as an extension to XORP

certain policy objectives through the simple AHP-based interface.

3.5 Implementation

We have implemented a Morpheus prototype as an extension to the XORP software router platform [53]. We first highlight the major changes we made to XORP, then describe the four policy classifiers and the decision process we have implemented in greater detail.

3.5.1 Changes to XORP

We chose XORP as the base platform to implement our Morpheus prototype because its modular structure closely parallels the Morpheus software architecture. However, since XORP is designed to implement the standard BGP decision process operating at a single router, our prototype differs from XORP's BGP implementation in three key ways.

First, we implemented the weighted-sum-based decision process of Morpheus from scratch. It has the ability to select different routes for different edge routers/peers, and can simultaneously run multiple decision processes each having its own policy configuration.

Second, to demonstrate that a policy classifier is easy to implement and to evaluate the performance of different such classifiers in action, we implemented four policy classifiers performing classifications based on business relationships, latency, stability and security respectively. While these classifiers could, in principle, work in parallel, in our prototype we implemented them as new modules in the XORP message processing pipeline, as shown in Figure 3.7. Since the classifiers

work independently, the ordering amongst them is not critical.

Third, we modified XORP's import and export-policy modules to bypass route-flap damping, and ensure export consistency between edge routers and the neighboring domains connected to them.

3.5.2 Policy Classifiers

In this section, we discuss in detail the four policy classifiers we have implemented thus far.

Business relationships: The business relationship classifier is linked to a configuration file that contains a table of (next-hop AS, business relationship) pairs. When a Morpheus server is started, it reads this file into memory. When a new route arrives, the classifier consults the table and assigns the route with the appropriate tag (e.g., "customer", "peer", or "provider").

Latency: Our latency classifier assumes there is a performance monitoring system (PMS) from which it periodically pulls real-time latency information about paths between an ingress point to an egress point, and from an egress link to a destination prefix. The retrieval of the performance information is handled by a background process and the pulling interval can be adjusted to reach a sweet spot between the freshness of the latency/loss information and the communication overhead to the PMS.

The latency classifier generates two types of tags—the absolute latency and the relative latency, to serve different policy needs—some policies only care about the relative latency amongst alternative paths (e.g., "always choose the path with the lowest latency"), while others may be specific about absolute latency (e.g., "for all paths with latency less than 100 ms, choose the most stable one through a customer"). To generate both types of tags, the latency classifier internally keeps records of the current path latency t_{now} and the minimum observed path latency t_{min} for each (prefix, next-hop) pair. When a new route arrives, it is tagged with t_{now} in milliseconds as the absolute latency, and t_{now}/t_{min} as the relative latency.

Stability: Our stability classifier implements the same penalty function as route flap damping does [98]. However, instead of suppressing routes with a penalty exceeding a threshold, our stability module tags each route with a penalty score.

Security: Our security classifier implements Pretty Good BGP (PGBGP) [59], a simple yet effective heuristic algorithm that identifies bogus routes based on a history of (prefix, origin AS) pairs. A route is tagged as "suspicious" if a route's AS path does not match the history of the last h days (where h is a configurable parameter); or as "unsuspicious" otherwise. This classifier is ported by the author of PGBGP from his original implementation, with a few interface changes. This demonstrates that the design of Morpheus is friendly to third-party modules.

Amongst the four classifiers, three of them (except the business-relationships classifier) are required to "re-tag" previously tagged routes when certain conditions are met. For example, the latency classifier needs to re-tag a route if the change in path latency exceeds a certain threshold. The stability classifier needs to re-tag a route when the decay of its penalty score exceeds certain value. The PGBGP algorithm also requires to re-tag a "suspicious" route as "unsuspicious" if it is not withdrawn after 24 hours. In all such cases, a configurable minimum re-tagging interval can be set to prevent undesirable flapping effect. (The 24-hour interval in the PGBGP case is long enough, so no additional constraint is needed.)

3.5.3 Decision Processes

We implemented the decision process with four mapping functions for the four classifiers, and a weighted-sum score function, as described in Section 3.3. Our implementation assumes the mapping functions and the score functions are specified in configuration files ahead of time. When a new route arrives, a decision process only computes the score for this new route, without re-calculating the scores for all previously received routes for the same prefix. In fact, the decision process only compares the new route's final score with the current *highest* score of that prefix. On

the other hand, when the current best route is withdrawn, the decision process compares the scores of all remaining routes and picks the one with the highest score as the new best route.

It is possible that more than one route receives the same score. To select a single best route for each peer/edge router in that case, Morpheus currently supports two types of tie-breaking mechanisms—ranking of egress points, and router ID. In the rank-based tie-breaking scheme, each edge router is assigned with a fixed (but configurable) ranking of all egress points. This ranking may reflect geographic distance or the typical IGP distances and link capacities between each pair of ingress/egress points. By decoupling changes in the IGP distances from the decision processes, the fixed-ranking scheme avoids the problems associated with hot-potato routing [93] and gives the ISP additional control over the flow of traffic (e.g., decrease an ingress point's ranking of a particular egress point, if a link gets overloaded by the traffic from the ingress point to that egress point). A closer coupling with the IGP distances, where needed, can be achieved on a longer time scale by simply adjusting the configuration of the fixed ranking.

3.6 Implementation and Evaluation

In this section, we evaluate the performance and scalability of Morpheus using our XORP-based prototype. Specifically, we answer three questions:

1. *What is the performance of Morpheus' policy classifiers and its score-based decision process?* We find that the Morpheus classifiers and decision process work efficiently. The average decision time of Morpheus is only 20% of the average time the standard BGP decision process takes, when there are 20 routes per prefix.

2. *Can Morpheus keep up with the rate of BGP update messages in large ISPs?* Our unoptimized prototype is able to achieve a sustained throughput of 890 updates/s, while the aggregated update arrival rate of a large tier-1 ISP is typically no larger than 600 updates/s [97].

3. *How many different policies (i.e., decision process instances) can Morpheus support efficiently?* Our experimental results show that our prototype can support 40 concurrent decision processes while achieving a sustainable throughput of 740 updates/s.

3.6.1 Evaluation Testbed

We conduct our experiments on a three-node testbed, consisting of an update generator, a Morpheus server, and an update receiver, interconnected through a switch. For a realistic evaluation, the route generator replays the RIB dump from RouteViews on April 17, 2007 [84] to the Morpheus server. The evaluations were performed with the Morpheus server and the update generator running on 3.2GHz Intel Pentium-4 platforms with 3.6GB of memory. We run the update receiver on a 2.8GHz Pentium-4 platform with 1GB of memory. The three machines each has one Gigabit Ethernet card and are connected through a Gigabit switch. They all run Linux 2.6.11 kernel.

3.6.2 Evaluation of Processing Time

To evaluate the performance of Morpheus' policy classifiers and decision process, we conduct white-box testing by instrumenting the classifier functions and the decision process, and measuring the time they take to process a route. To highlight the performance difference introduced by the Morpheus design, we also compare Morpheus' decision time with two reference implementations in XORP: the standard BGP decision process and a modified BGP decision process with a rank-based tie-breaking step [4] (similar to what Morpheus uses) after the multi-exit discriminator (MED) comparison step. In each processing-time experiment, the update generator sends 100,000 updates to the Morpheus server.

Classification time: We first measure the time each policy classifier takes to tag a route. In this experiment, the business-relationship classifier reads in a table of 2000 (AS number, business rela-

[4] In the rank-based tie-breaking scheme, each edge router is assigned with a fixed (but configurable) ranking of all egress points, and the edge router with the highest ranking is selected as the winner [102].

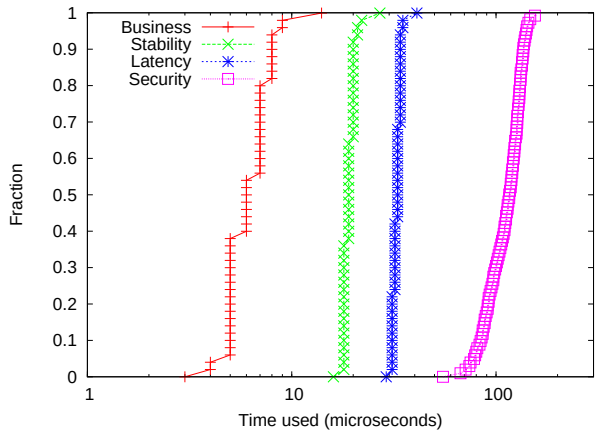


Figure 3.8: Classification time: time taken by the classifiers to tag a route.

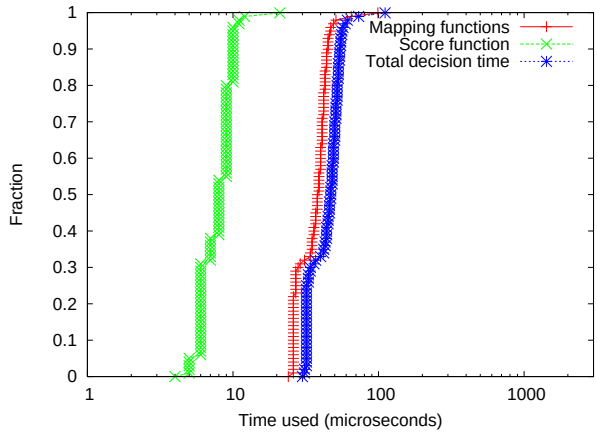


Figure 3.9: Decision time: time taken by the mapping functions and the score function, and the total decision time (1 route per prefix)

tionship) pairs. The latency classifier is fed with static tables of path latency data. We believe the result we get should be comparable to the scenario in which Morpheus gets this information from a monitoring system, because the measurement results will be pre-fetched by a background process and cached. From the CDF of the tagging time shown in Figure 3.8, we see that the business-relationship classifier takes only about 5 microseconds to tag a route. The stability classifier takes about 20 microseconds on average, while the delay classifier takes about 33 microseconds. The most complex classifier—the security classifier which implements the PG-BGP algorithm, takes 103 microseconds on average.

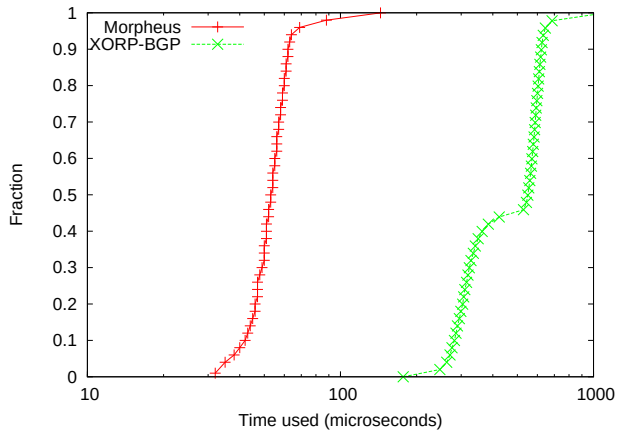


Figure 3.10: Decision time: comparison between Morpheus and XORP-BGP, 20 routes per prefix.

Decision time (one route per prefix): We then benchmark the time taken by the decision process to calculate the final score for a route (excluding the classification time). Figure 3.9 shows the CDFs of the two components of the decision time—the mapping functions (one for each classifier) and the score function, as well as the total time. As we expected, the score function runs very quickly, taking only 8 microseconds on average. The four mapping functions take 37 microseconds in total. The total decision time is about 45 microseconds on average. In this experiment, the update generator only sends one update per prefix to the Morpheus server, so there is no tie-breaking involved in our measurements.

Decision time (multiple alternative routes per prefix): In the next experiment, we compare the decision time of Morpheus and the out-of-the-box BGP implementation of XORP (XORP-BGP), when each prefix has multiple alternative routes. We configure both Morpheus and XORP-BGP to receive 20 identical (except for router IDs) routes per prefix from the update generator. To make a fair comparison, we configure Morpheus to use router ID to break ties. From Figure 3.10 we can see Morpheus takes about 54 microseconds on average to select a best route, whereas XORP-BGP takes an average time of 279 microseconds.

It is not surprising to see that Morpheus takes much less time than XORP-BGP in selecting best route when the number of alternative routes is large, because regardless of the number of alternative

routes per prefix, Morpheus only needs to compute one score when a new route arrives, whereas XORP-BGP has to compare the pool of alternative routes for the same prefix all together through the step-by-step comparisons in the BGP decision process. This also explains why the decision time of Morpheus has smaller variation, while XORP-BGP's decision time varies significantly, ranging from less than 100 microseconds (when there is only a small number of alternative routes for a prefix) to over 500 microseconds (when the number becomes large).

Table 3.2: Processing time of the rank-based tie-breaker

	10 routes/prefix	20 routes/prefix
10 edge routers	83 μs	175 μs
20 edge routers	138 μs	309 μs

Time to perform rank-based tie-breaking: Finally we measure the time Morpheus takes to perform rank-based tie-breaking when multiple alternative routes have the same score. Without any knowledge about how often and how many routes will end up having the same score, we study two cases in our experiments: the *random case* and the *worst case*. In the random case, we assign every alternative route with a random integer score uniformly selected between 0 and 100. In the worst case, we let all alternative routes per prefix have the same score. We run eight test cases: random case/worst case with 10/20 edge routers and with 10/20 routes per prefix. Since in the four random cases, there is little possibility (i.e., $\binom{20}{2} \cdot 0.01^2 = 0.019$) that two routes will have the same final score, leaving the rank-based tie-breaker almost never used, we list only the average tie-breaking time of the four worst cases in Table 3.2. As we can see, if *all* alternative routes happen to have the same score, the rank-based tie-breaking step will become the performance bottleneck of Morpheus' decision process, even in the modest case of 10 routes/prefix with 10 edge routers. However, such worst case scenario is not likely to happen very often in reality, especially when the number of alternative routes is relatively large.

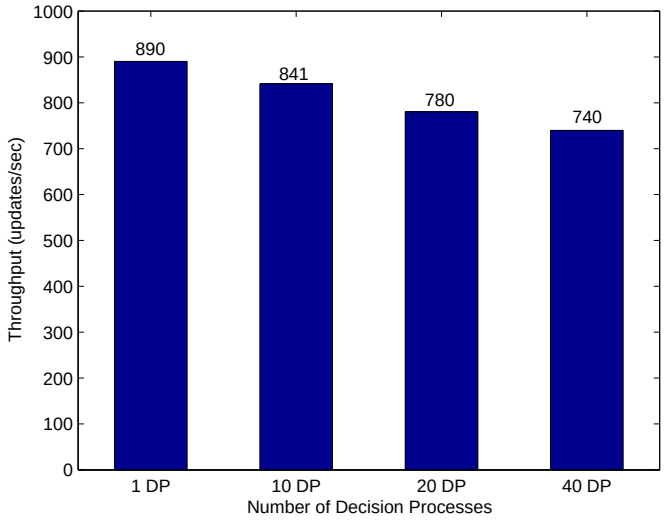


Figure 3.11: Throughput achieved by Morpheus with different number of decision processes

3.6.3 Throughput

To determine the update processing throughput Morpheus can achieve, we use the following network model of a large tier-1 ISP from a prior study [97]. We assume a large ISP has 40 Point-of-Presence (POPs), each of which contains one Morpheus server. Each Morpheus server has 240 eBGP sessions with customers, and 15 iBGP sessions with edge routers. It also keeps 40 sessions with other Morpheus servers, through which it learns every route other Morpheus server receives. We assume the ISP (and each of its Morpheus servers) receives 20 routes per prefix (as shown in Section 2.1). Since each Morpheus server selects routes for the edge routers located the same POP, in the experiments we assume it applies the same ranking of egress points for all its 15 edge routers, while different Morpheus servers still have different rankings.

In each throughput experiment, the update generator maintains 20 sessions with the Morpheus server and sends 20 update messages of each route, one per session. The Morpheus server maintains 295 sessions (240 eBGP sessions, 15 iBGP sessions and 40 sessions to other Morpheus servers) with the update receiver. By sending multiple routes with the same attributes to the Morpheus server, we measure the *worst case* throughput Morpheus can achieve, because all the routes

will have the same score and hence the rank-based tie-breaking step is always performed. Every throughput experiment runs for a 15-minutes period, and the update generator sends updates at the fastest rate it can get.

Figure 3.11 compares the throughput achieved by Morpheus configured with different number of decision processes. When Morpheus only runs one decision process, it achieves a sustained throughput of 890 updates/s. As the number of decision processes increases to 10, 20 and 40, the achieved throughput decreases slowly to 841, 780 and 740 updates/s, respectively. When we increase the number of decision processes, we assume each customer still subscribes to only one of them (i.e., only receives one route per prefix). As such, the total number of updates the Morpheus sends to the update receiver does not increase.

We are satisfied with the throughput our unoptimized prototype achieves, as a large tier-1 ISP usually receives less than 600 updates/s (95 percentile) [97]. The slow decrease of throughput as the number of decision processes increases also demonstrates Morpheus' score-based decision process design can scale to a large number of different policies.

3.6.4 Memory Requirement

When we compare the memory consumption of Morpheus with XORP-BGP, we find XORP-BGP consumes 970 MB of memory when loaded with five routes per prefix, as its implementation stores multiple copies of each route. Therefore, neither the out-of-box XORP-BGP nor our XORP-based Morpheus prototype were able to load 20 full BGP routing tables with 3.6 GB of memory. However, the memory footprint of our Morpheus prototype is only 10% larger than that of XORP-BGP, which is mainly used by the classifiers (largely by the security classifier) and used to store metadata of routes (tags, scores, etc.). We observe that other BGP implementations consume much less memory comparing to XORP under the same condition. For example, openbgpd only takes 270 MB to store 20 full BGP routing tables [97]. Our experiment on Quagga [79] shows a memory footprint of 550 MB with 20 full BGP routing tables. Therefore, we believe that a Morpheus

prototype based on a BGP implementation with better memory efficiency will not impose any memory pressure on a reasonably provisioned server. We also note that, unlike router memory, memory for regular servers is cheap and easy to install.

3.7 Related Work

Previous work proposes to raise the level of abstraction of BGP policy configuration through network-wide, vendor-neutral specification languages [5, 12]. However, we believe new languages alone are not sufficient to make policy configuration more flexible, because today's intra-AS routing architecture and the current BGP decision process both introduce peculiar constraints on the set of policies that can be realized. In this chapter, we take a fresh approach of "design for configurability" and present a system that supports more flexible routing policies and yet is easier to configure.

Several recent studies on the Routing Control Platform (RCP) [33] advocate moving the BGP control plane of a single AS to a small set of servers that select routes on behalf of the routers [15, 96, 97, 6]. The prototype systems in [15] and [97] demonstrate that a logically-centralized control plane running on commodity hardware can be scalable, reliable, and fast enough to drive BGP routing decisions in a large ISP backbone. However, the system in [15] simply mimics the standard BGP decision process, without expanding the space of realizable policies. While [96] and [97] support more flexible alternatives to today's hot-potato routing, these systems do not create an extensible framework for realizing flexible policies with trade-offs amongst policy objectives, or support multiple different policies simultaneously. They do not revisit the convoluted BGP configuration interface either. These are the main contributions of our Morpheus design.

3.8 Summary

This chapter presents the design, implementation and evaluation of Morpheus, a routing control platform that enables a single ISP to realize many useful routing policies that are infeasible today without changing its routers. The design of the Morpheus server separates route classification from route selection, which enables network operators to easily define new policy objectives, implement independent objective classifiers, and make flexible trade-offs between objectives. Morpheus allows large ISPs to capitalize on their path diversity and provide customer-specific routes as a value-added service. It also enables an ISP to allow its customers to influence its routing policies through a simple and intuitive configuration interface. Our experiments show that Morpheus can support a large number of different policies simultaneously while handling the high rate of BGP updates experienced in large ISPs.

Chapter 4

VROOM: Live (Virtual) Router Migration as a Network-Management Primitive

4.1 Introduction

In Chapters 2 and 3, we focused on how an ISP can realize flexible and customizable routing policies through intuitive configuration interface. To ensure quality of service, besides such *active* management of routing in its network, an ISP also needs to make sure that other network management operations introduce as little disruption to routing as possible. However, from routine tasks such as planned maintenance to the less-frequent deployment of new protocols, network operators struggle to provide seamless service in the face of changes to the underlying network.

Handling change is difficult because each change to the physical infrastructure requires a corresponding modification to the logical configuration of the routers—such as reconfiguring the tunable parameters in the routing protocols. *Logical* refers to IP packet-forwarding functions, while *physical* refers to the physical router equipment (such as line cards and the CPU) that enables these functions. Any inconsistency between the logical and physical configurations can lead to unexpected reachability or performance problems. Furthermore, because of today's tight coupling

between the physical and logical topologies, logical-layer changes are sometimes used purely as a *tool* to handle physical changes more gracefully. A classic example is increasing the link weights in Interior Gateway Protocols to "cost out" a router in advance of planned maintenance [93]. In this case, a change in the logical topology is *not* the goal, rather it is the indirect tool available to achieve the task at hand, and it does so with potential negative side effects.

In this chapter, we argue that breaking the tight coupling between physical and logical configurations can provide a *single*, general abstraction that simplifies network management. Specifically, we propose VROOM (Virtual ROuters On the Move), a new network-management primitive where virtual routers can move freely from one physical router to another. In VROOM, physical routers merely serve as the carrier substrate on which the actual virtual routers operate. VROOM can migrate a virtual router to a different physical router without disrupting the flow of traffic or changing the logical topology, obviating the need to reconfigure the virtual routers while also avoiding routing-protocol convergence delays. For example, if a physical router must undergo planned maintenance, the virtual routers could move (in advance) to another physical router in the same Point-of-Presence (PoP). In addition, edge routers can move from one location to another by virtually re-homing the links that connect to neighboring domains.

Realizing these objectives presents several challenges: (i) *migratable routers*: to make a (virtual) router migratable, its "router" functionality must be separable from the physical equipment on which it runs; (ii) *minimal outages:* to avoid disrupting user traffic or triggering routing protocol reconvergence, the migration should cause no or minimal packet loss; (iii) *migratable links:* to keep the IP-layer topology intact, the links attached to a migrating router must "follow" it to its new location. Fortunately, the third challenge is addressed by recent advances in transport-layer technologies, as discussed in Section 4.2. Our goal, then, is to migrate router functionality from one piece of equipment to another without disrupting the IP-layer topology or the data traffic it carries, and without requiring router reconfiguration.

On the surface, virtual router migration might seem like a straight-forward extention to existing

virtual machine migration techniques. This would involve copying the virtual router image (including routing-protocol binaries, configuration files and data-plane state) to the new physical router and freezing the running processes before copying them as well. The processes and data-plane state would then be restored on the new physical router and associated with the migrated links. However, the delays in completing all of these steps would cause unacceptable disruptions for both the data traffic and the routing protocols. For virtual router migration to be viable in practice, packet forwarding should not be interrupted, not even temporarily. In contrast, the control plane can tolerate brief disruptions, since routing protocols have their own retransmission mechansisms. Still, the control plane must restart quickly at the new location to avoid losing protocol adjacencies with other routers and to minimize delay in responding to unplanned network events.

In VROOM, we minimize disruption by leveraging the separation of the control and data planes in modern routers. We introduce a *data-plane hypervisor*—a migration-aware interface between the control and data planes. This unified interface allows us to support migration between physical routers with different data-plane technologies. VROOM migrates only the control plane, while continuing to forward traffic through the old data plane. The control plane can start running at the new location, and populate the new data plane while updating the old data plane in parallel. During the transition period, the old router redirects routing-protocol traffic to the new location. Once the data plane is fully populated at the new location, link migration can begin. The two data planes operate simultaneously for a period of time to facilitate asynchronous migration of the links.

To demonstrate the generality of our data-plane hypervisor, we present two prototype VROOM routers—one with a software data plane (in the Linux kernel) and the other with a hardware data plane (using a NetFPGA card [72]). Each virtual router runs the Quagga routing suite [79] in an OpenVZ container [73]. Our software extensions consist of three main modules that (i) separate the forwarding tables from the container contexts, (ii) push the forwarding-table entries generated by Quagga into the separate data plane, and (iii) dynamically bind the virtual interfaces and forwarding tables. Our system supports seamless live migration of virtual routers between the two

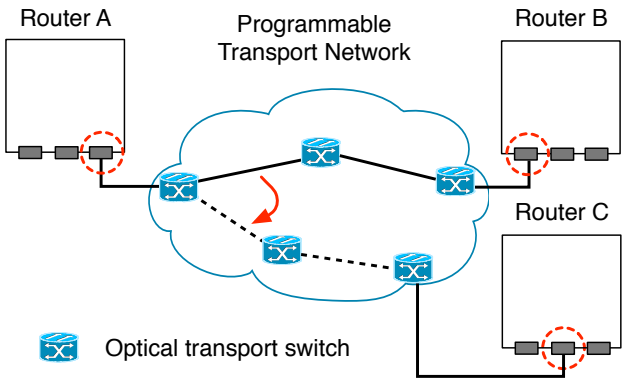
data-plane platforms. Our experiments show that virtual router migration causes no packet loss or delay when the hardware data plane is used, and at most a few seconds of delay in processing control-plane messages.

The remainder of the chapter is structured as follows. Section 4.2 presents background on flexible transport networks and an overview of related work. Next, Section 4.3 discusses how router migration would simplify existing network management tasks, such as planned maintenance and service deployment, while also addressing emerging challenges like power management. We present the VROOM architecture in Section 4.4, followed by the implementation and evaluation in Sections 4.5 and 4.6, respectively. We discuss migration scheduling in Section 4.7 and related work in Section 4.8, before summarizing this chapter in Section 4.9.

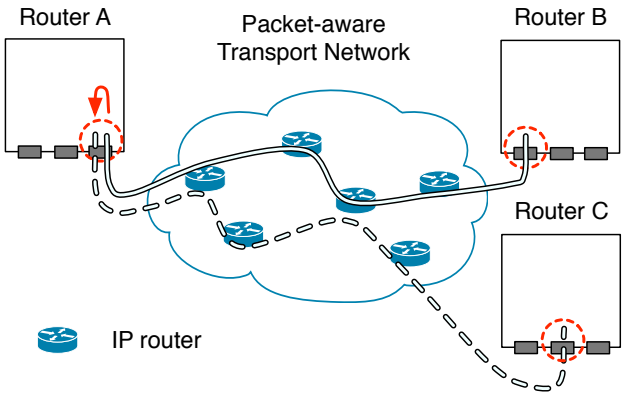
4.2 Background: Flexible Link Migration

One of the fundamental requirements of VROOM is "link migration", i.e., the links of a virtual router should "follow" its migration from one physical node to another. This is made possible by emerging transport network technologies.

In its most basic form, a link at the IP layer corresponds to a direct physical link (e.g., a cable), making link migration hard as it involves physically moving link end point(s). However, in practice, what appears as a direct link at the IP layer often corresponds to a series of connections through different network elements at the transport layer. For example, in today's ISP backbones, "direct" physical links are typically realized by optical transport networks, where an IP link corresponds to a circuit traversing multiple optical switches [22, 107]. Recent advances in *programmable transport networks* [22, 4] allow physical links between routers to be dynamically set up and torn down. For example, as shown in Figure 4.1(a), the link between physical routers A and B is switched through a programmable transport network. By signaling the transport network, the same physical port on router A can be connected to router C after an optical path switch-over.



(a) Programmable transport network



(b) Packet-aware transport network

Figure 4.1: Link migration in the transport networks

Such path switch-over at the transport layer can be done efficiently, e.g., sub-nanosecond optical switching time has been reported [82]. Furthermore, such switching can be performed across a wide-area network of transport switches, which enables inter-POP link migration.

In addition to *core links* within an ISP, we also want to migrate *access links* connecting customer edge (CE) routers and provider edge (PE) routers, where only the PE end of the links are under the ISP's control. Historically, access links correspond to a path in the underlying access network, such as a T1 circuit in a time-division multiplexing (TDM) access network. In such cases, the migration of an access link can be accomplished in similar fashion to the mechanism shown in Figure 4.1(a), by switching to a new circuit at the switch directly connected to the CE router. However, in traditional circuit-switched access networks, a dedicated physical port on a PE router is required to terminate each TDM circuit. Therefore, if all ports on a physical PE router are in use, it will not be able to accommodate more virtual routers. Fortunately, as Ethernet emerges as an economical and flexible alternative to legacy TDM services, access networks are evolving to *packet-aware* transport networks [3]. This trend offers important benefits for VROOM by eliminating the need for per-customer physical ports on PE routers. In a packet-aware access network (e.g., a virtual private LAN service access network), each customer access port is associated with a label, or a "pseudo wire" [14], which allows a PE router to support multiple logical access links on the same physical port. The migration of a pseudo-wire access link involves establishing a new pseudo wire and switching to it at the multi-service switch [3] adjacent to the CE.

Unlike conventional ISP networks, some networks are realized as overlays on top of other ISPs' networks. Examples include commercial "Carrier Supporting Carrier (CSC)" networks [24], and VINI, a research virtual network infrastructure overlaid on top of National Lambda Rail and Internet2 [99]. In such cases, a single-hop link in the overlay network is actually a multi-hop path in the underlying network, which can be an MPLS VPN (e.g., CSC) or an IP network (e.g., VINI). Link migration in an MPLS transport network involves switching over to a newly established label switched path (LSP). Link migration in an IP network can be done by changing the IP address of

the tunnel end point.

4.3 Network Management Tasks

In this section, we present three case studies of the applications of VROOM. We show that the separation between physical and logical, and the router migration capability enabled by VROOM, can greatly simplify existing network-management tasks. It can also provide network-management solutions to other emerging challenges. We explain why the existing solutions (in the first two examples) are not satisfactory and outline the VROOM approach to addressing the same problems.

4.3.1 Planned Maintenance

Planned maintenance is a hidden fact of life in every network. However, the state-of-the-art practices are still unsatisfactory. For example, software upgrades today still require rebooting the router and re-synchronizing routing protocol states from neighbors (e.g., BGP routes), which can lead to outages of 10-15 minutes [4]. Different solutions have been proposed to reduce the impact of planned maintenance on network traffic, such as "costing out" the equipment in advance. Another example is the RouterFarm approach of removing the static binding between customers and access routers to reduce service disruption time while performing maintenance on access routers [4]. However, we argue that neither solution is satisfactory, since maintenance of *physical* routers still requires changes to the *logical* network topology, and requires (often human interactive) reconfigurations and routing protocol reconvergence. This usually implies more configuration errors [60] and increased network instability.

We performed an analysis of planned-maintenance events conducted in a Tier-1 ISP backbone over a one-week period. Due to space limitations, we only mention the high-level results that are pertinent to VROOM here. Our analysis indicates that, among all the planned-maintenance events that have undesirable network impact today (e.g., routing protocol reconvergence or data-

plane disruption), 70% could be conducted without any network impact if VROOM were used. (This number assumes migration between routers with control planes of like kind. With more sophisticated migration strategies, e.g., where a "control-plane hypervisor" allows migration between routers with different control plane implementations, the number increases to 90%.) These promising numbers result from the fact that most planned-maintenance events were hardware related and, as such, did not intend to make any longer-term changes to the logical-layer configurations.

To perform planned maintenance tasks in a VROOM-enabled network, network administrators can simply migrate all the virtual routers running on a physical router to other physical routers before doing maintenance and migrate them back afterwards as needed, without ever needing to reconfigure any routing protocols or worry about traffic disruption or protocol reconvergence.

4.3.2 Service Deployment and Evolution

Deploying new services, like IPv6 or IPTV, is the life-blood of any ISP. Yet, ISPs must exercise caution when deploying these new services. First, they must ensure that the new services do not adversely impact existing services. Second, the necessary support systems need to be in place before services can be properly supported. (Support systems include configuration management, service monitoring, provisioning, and billing.) Hence, ISPs usually start with a small trial running in a controlled environment on dedicated equipment, supporting a few early-adopter customers. However, this leads to a "success disaster" when the service warrants wider deployment. The ISP wants to offer seamless service to its existing customers, and yet also restructure their test network, or move the service onto a larger network to serve a larger set of customers. This "trial system success" dilemma is hard to resolve if the *logical* notion of a "network node" remains bound to a specific *physical* router.

VROOM provides a simple solution by enabling network operators to freely migrate virtual routers from the trial system to the operational backbone. Rather than shutting down the trial service, the ISP can continue supporting the early-adopter customers while continuously growing

the trial system, attracting new customers, and eventually seamlessly migrating the entire service to the operational network.

ISPs usually deploy such service-oriented routers as close to their customers as possible, in order to avoid backhaul traffic. However, as the services grow, the geographical distribution of customers may change over time. With VROOM, ISPs can easily reallocate the routers to adapt to new customer demands.

4.3.3 Power Savings

VROOM not only provides simple solutions to conventional network-management tasks, but also enables new solutions to emerging challenges such as power management. It was reported that in 2000 the total power consumption of the estimated 3.26 million routers in the U.S. was about 1.1 TWh (Tera-Watt hours) [83]. This number was expected to grow to 1.9 to 2.4TWh in the year 2005 [83], which translates into an annual cost of about 178-225 million dollars [78]. These numbers do not include the power consumption of the required cooling systems.

Although designing energy-efficient equipment is clearly an important part of the solution [52], we believe that network operators can also *manage* a network in a more power-efficient manner. Previous studies have reported that Internet traffic has a consistent diurnal pattern caused by human interactive network activities. However, today's routers are surprisingly power-insensitive to the traffic loads they are handling—an idle router consumes over 90% of the power it requires when working at maximum capacity [17]. We argue that, with VROOM, the variations in daily traffic volume can be exploited to reduce power consumption. Specifically, the size of the physical network can be expanded and shrunk according to traffic demand, by hibernating or powering down the routers that are not needed. The best way to do this today would be to use the "cost-out/cost-in" approach, which inevitably introduces configuration overhead and performance disruptions due to protocol reconvergence.

VROOM provides a cleaner solution: as the network traffic volume decreases at night, virtual

routers can be migrated to a smaller set of physical routers and the unneeded physical routers can be shut down or put into hibernation to save power. When the traffic starts to increase, physical routers can be brought up again and virtual routers can be migrated back accordingly. With VROOM, the IP-layer topology stays intact during the migrations, so that power savings do not come at the price of user traffic disruption, reconfiguration overhead or protocol reconvergence. Our analysis of data traffic volumes in a Tier-1 ISP backbone suggests that, even if only migrating virtual routers within the same POP while keeping the same link utilization rate, applying the above VROOM power management approach could save 18%-25% of the power required to run the routers in the network. As discussed in Section 4.7, allowing migration across different POPs could result in more substantial power savings.

4.4 VROOM Architecture

In this section, we present the VROOM architecture. We first describe the three building-blocks that make virtual router migration possible—router virtualization, control and data plane separation, and dynamic interface binding. We then present the VROOM router migration process. Unlike regular servers, modern routers typically have physically separate control and data planes. Leveraging this unique property, we introduce a *data-plane hypervisor* between the control and data planes that enables virtual routers to migrate across different data-plane platforms. We describe in detail the three migration techniques that minimize control-plane downtime and eliminate data-plane disruption—data-plane cloning, remote control plane, and double data planes.

4.4.1 Making Virtual Routers Migratable

Figure 4.2 shows the architecture of a VROOM router that supports virtual router migration. It has three important features that make migration possible: router virtualization, control and data plane separation, and dynamic interface binding, all of which already exist in some form in today's

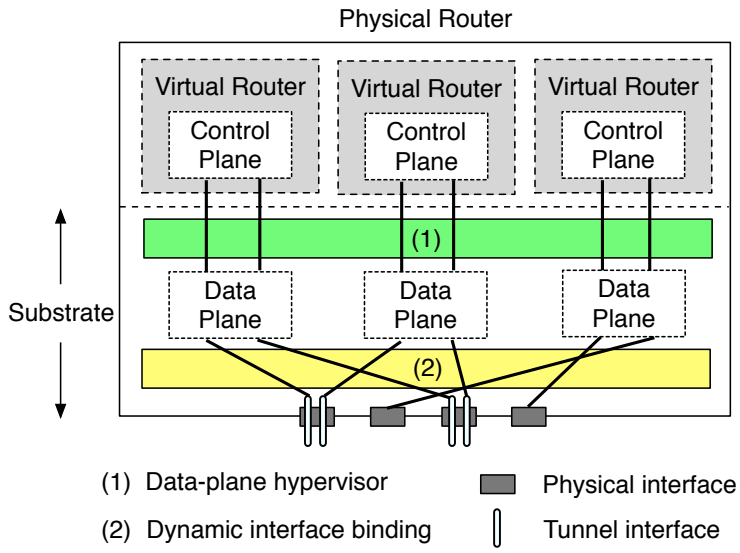


Figure 4.2: The architecture of a VROOM router

high-end commercial routers.

Router Virtualization: A VROOM router partitions the resources of a physical router to support multiple *virtual router* instances. Each virtual router runs independently with its own control plane (e.g., applications, configurations, routing protocol instances and routing information base (RIB)) and data plane (e.g., interfaces and forwarding information base (FIB)). Such *router virtualization* support is already available in some commercial routers [25, 57]. The isolation between virtual routers makes it possible to migrate one virtual router without affecting the others.

Control and Data Plane Separation: In a VROOM router, the control and data planes run in *separate* environments. As shown in Figure 4.2, the control planes of virtual routers are hosted in separate "containers" (or "virtual environments"), while their data planes reside in the *substrate*, where each data plane is kept in separate data structures with its own state information, such as FIB entries and access control lists (ACLs). Similar separation of control and data planes already exists in today's commercial routers, with control plane running on the CPU(s) and main memory, while the data plane runs on line cards that have their own computing power (for packet forwarding) and memory (to hold the FIBs). This separation allows VROOM to migrate the control and data planes

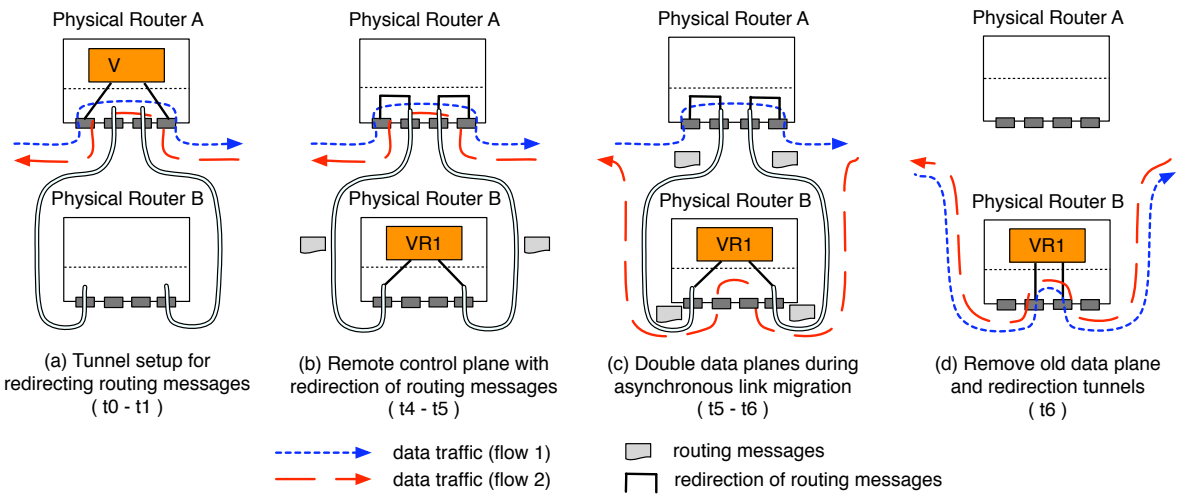


Figure 4.3: VROOM's novel router migration mechanisms (the times at the bottom of the subfigures correspond to those in Figure 4.4)

of a virtual router separately (as discussed in Section 4.4.2 and 4.4.2).

Dynamic Interface Binding: To enable router migration and link migration, a VROOM router should be able to *dynamically* set up and change the binding between a virtual router's FIB and its *substrate interfaces* (which can be physical or tunnel interfaces), as shown in Figure 4.2. Given the existing interface binding mechanism in today's routers that maps interfaces with virtual routers, VROOM only requires two simple extensions. First, after a virtual router is migrated, this binding needs to be re-established dynamically on the new physical router. This is essentially the same as if this virtual router were just instantiated on the physical router. Second, link migration in a packet-aware transport network involves changing tunnel interfaces in the router, as shown in Figure 4.1. In this case, the router substrate needs to switch the binding from the old tunnel interface to the new one on-the-fly[1].

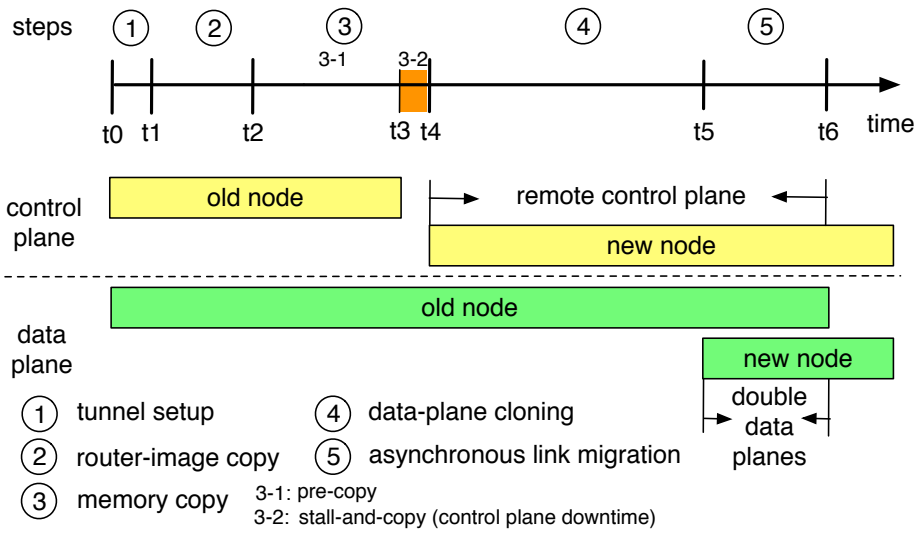


Figure 4.4: VROOM's router migration process

4.4.2 Virtual Router Migration Process

Figures 4.3 and 4.4 illustrate the VROOM virtual router migration process. The first step in the process involves establishing tunnels between the source physical router A and destination physical router B of the migration (Figure 4.3(a)). These tunnels allow the control plane to send and receive routing messages after it is migrated (steps 2 and 3) but before link migration (step 5) completes. They also allow the migrated control plane to keep its data plane on A up-to-date (Figure 4.3(b)). Although the control plane will experience a short period of downtime at the end of step 3 (memory copy), the data plane continues working during the entire migration process. In fact, after step 4 (data-plane cloning), the data planes on both A and B can forward traffic simultaneously (Figure 4.3(c)). With these double data planes, links can be migrated from A to B in an asynchronous fashion (Figure 4.3(c) and (d)), after which the data plane on A can be disabled (Figure 4.4). We now describe the migration mechanisms in greater detail.

[1]In the case of a programmable transport network, link migration happens inside the transport network and is transparent to the routers.

Control-Plane Migration

Two things need to be taken care of when migrating the control plane: the *router image*, such as routing-protocol binaries and network configuration files, and the *memory*, which includes the states of all the running processes. When copying the router image and memory, it is desirable to minimize the total migration time, and more importantly, to minimize the control-plane downtime (i.e., the time between when the control plane is check-pointed on the source node and when it is restored on the destination node). This is because, although routing protocols can usually tolerate a brief network glitch using retransmission (e.g., BGP uses TCP retransmission, while OSPF uses its own reliable retransmission mechanism), a long control-plane outage can break protocol adjacencies and cause protocols to reconverge.

We now describe how VROOM leverages virtual machine (VM) migration techniques to migrate the control plane in steps 2 (router-image copy) and 3 (memory copy) of its migration process, as shown in Figure 4.4.

Unlike general-purpose VMs that can potentially be running completely different programs, virtual routers from the same vendor run the same (usually small) set of programs (e.g., routing protocol suites). VROOM assumes that the same set of binaries are already available on every physical router. Before a virtual router is migrated, the binaries are locally copied to its file system on the destination node. Therefore, only the router configuration files need to be copied over the network, reducing the total migration time (as local-copy is usually faster than network-copy).

The simplest way to migrate the memory of a virtual router is to check-point the router, copy the memory pages to the destination, and restore the router, a.k.a. *stall-and-copy* [73]. This approach leads to downtime that is proportional to the memory size of the router. A better approach is to add an iterative *pre-copy* phase before the final stall-and-copy [27], as shown in Figure 4.4. All pages are transferred in the first round of the pre-copy phase, and in the following rounds, only pages that were modified during the previous round are transferred. This pre-copy technique reduces the number of pages that need to be transfered in the stall-and-copy phase, reducing the control

plane downtime of the virtual router (i.e., the control plane is only "frozen" between t3 and t4 in Figure 4.4).

Data-Plane Cloning

The control-plane migration described above could be extended to migrate the data plane, i.e., copy all data-plane states over to the new physical node. However, this approach has two drawbacks. First, copying the data-plane states (e.g., FIB and ACLs) is unnecessary and wasteful, because the information that is used to generate these states (e.g., RIB and configuration files) is already available in the control plane. Second, copying the data-plane state directly can be difficult if the source and destination routers use different data-plane technologies. For example, some routers may use TCAM (ternary content-addressable memory) in their data planes, while others may use regular SRAM. As a result, the data structures that hold the state may be different.

VROOM formalizes the interface between the control and data planes by introducing a *data-plane hypervisor*, which allows a migrated control plane to re-instantiate the data plane on the new platform, a process we call **data-plane cloning**. That is, only the control plane of the router is actually migrated. Once the control plane is migrated to the new physical router, it *clones* its original data plane by repopulating the FIB using its RIB and reinstalling ACLs and other data-plane states[2] through the data-plane hypervisor (as shown in Figure 4.2). The data-plane hypervisor provides a unified interface to the control plane that hides the heterogeneity of the underlying data-plane implementations, enabling virtual routers to migrate between different types of data planes.

Remote Control Plane

As shown in Figure 4.3(b), after VR1's control plane is migrated from A to B, the natural next steps are to repopulate (clone) the data plane on B and then migrate the links from A to B. Unfortunately,

[2]Data dynamically collected in the old data plane (such as NetFlow) can be copied and merged with the new one. Other path-specific statistics (such as queue length) will be reset as the previous results are no longer meaningful once the physical path changes.

the creation of the new data plane can not be done instantaneously, primarily due to the time it takes to install FIB entries. Installing one FIB entry typically takes between one hundred and a few hundred microseconds [13]; therefore, installing the full Internet BGP routing table (about 250k routes) could take over 20 seconds. During this period of time, although data traffic can still be forwarded by the old data plane on A, all the routing instances in VR1's control plane can no longer send or receive routing messages. The longer the control plane remains unreachable, the more likely it will lose its protocol adjacencies with its neighbors.

To overcome this dilemma, A's substrate starts redirecting all the routing messages destined to VR1 to B at the end of the control-plane migration (time t4 in Figure 4.4). This is done by establishing a tunnel between A and B for each of VR1's substrate interfaces. To avoid introducing any additional downtime in the control plane, these tunnels are established before the control-plane migration, as shown in Figure 4.3(a). With this redirection mechanism, VR1's control plane not only can exchange routing messages with its neighbors, it can also act as the **remote control plane** for its old data plane on A and continue to update the old FIB when routing changes happen.

Double Data Planes

In theory, at the end of the data-plane cloning step, VR1 can switch from the old data plane on A to the new one on B by migrating all its links from A to B simultaneously. However, performing accurate synchronous link migration across all the links is challenging, and could significantly increase the complexity of the system (because of the need to implement a synchronization mechanism).

Fortunately, because VR1 has **two** data planes ready to forward traffic at the end of the data-plane cloning step (Figure 4.4), the migration of its links does not need to happen all at once. Instead, each link can be migrated independent of the others, in an asynchronous fashion, as shown in Figure 4.3(c) and (d). First, router B creates a new *outgoing* link to each of VR1's neighbors, while all data traffic continues to flow through router A. Then, the *incoming* links can be safely migrated asynchronously, with some traffic starting to flow through router B while the remaining

traffic still flows through router A. Finally, once all of VR1's links are migrated to router B, the old data plane and outgoing links on A, as well as the temporary tunnels, can be safely removed.

4.5 Prototype Implementation

In this section, we present the implementation of two VROOM prototype routers. The first is built on commodity PC hardware and the Linux-based virtualization solution OpenVZ [73]. The second is built using the same software but utilizing the NetFPGA platform [72] as the hardware data plane. We believe the design presented here is readily applicable to commercial routers, which typically have the same clean separation between the control and data planes.

Our prototype implementation consists of three new programs, as shown in Figure 4.5. These include `virtd`, to enable packet forwarding outside of the virtual environment (control and data plane separation); `shadowd`, to enable each VE to install routes into the FIB; and `bindd` (data plane cloning), to provide the bindings between the physical interfaces and the virtual interfaces and FIB of each VE (data-plane hypervisor). We first discuss the mechanisms that enable virtual router migration in our prototypes and then present the additional mechanisms we implemented that realize the migration.

4.5.1 Enabling Virtual Router Migration

We chose to use OpenVZ [73], a Linux-based OS-level virtualization solution, as the virtualization environment for our prototypes. As running multiple operating systems for different virtual routers is unnecessary, the lighter-weight OS-level virtualization is better suited to our need than other virtualization techniques, such as full virtualization and para-virtualization. In OpenVZ, multiple virtual environments (VEs) running on the same host share the same kernel, but have separate virtualized resources such as name spaces, process trees, devices, and network stacks. OpenVZ

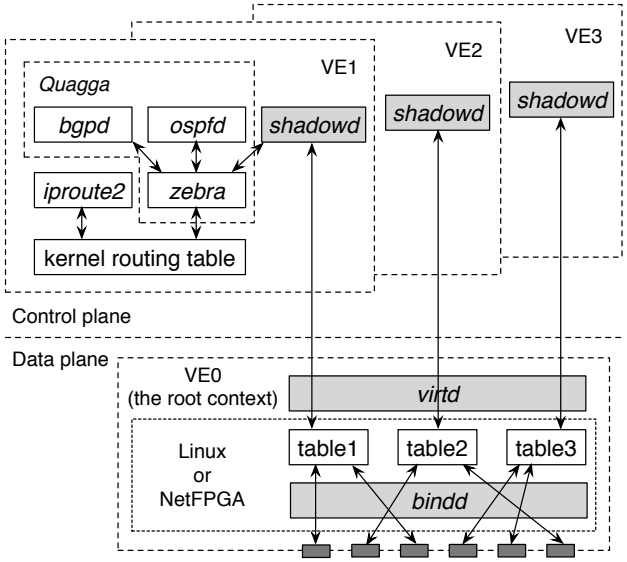


Figure 4.5: The design of the VROOM prototype routers (with two types of data planes)

also provides live migration capability for running VEs[3].

In the rest of this subsection, we describe in a top-down order the three components of our two prototypes that enable virtual router migration. We first present the mechanism that separates the control and data planes, and then describe the data-plane hypervisor that allows the control planes to update the FIBs in the shared data plane. Finally, we describe the mechanisms that dynamically bind the interfaces with the FIBs and set up the data path.

Control and Data Plane Separation

To mimic the control and data plane separation provided in commercial routers, we move the FIBs out of the VEs and place them in a shared but virtualized data plane, as shown in Figure 4.5. This means that packet forwarding no longer happens within the context of each VE, so it is unaffected when the VE is migrated.

As previously mentioned, we have implemented two prototypes with different types of data planes—a software-based data plane (SD) and a hardware-based data plane (HD). In the SD proto-

[3]The current OpenVZ migration function uses the simple "stall-and-copy" mechanism for memory migration. Including a "pre-copy" stage [27] in the process will reduce the migration downtime.

type router, the data plane resides in the root context (or "VE0") of the system and uses the Linux kernel for packet forwarding. Since the Linux kernel (2.6.18) supports 256 separate routing tables, the SD router virtualizes its data plane by associating each VE with a different kernel routing table as its FIB.

In the HD router implementation, we use the NetFPGA platform configured with the reference router provided by Stanford [72]. The NetFPGA card is a 4-port gigabit ethernet PCI card with a Virtex 2-Pro FPGA on it. With the NetFPGA as the data plane, packet forwarding in the HD router does not use the host CPU, thus more closely resembling commercial router architectures. The NetFPGA reference router does not currently support virtualization. As a result, our HD router implementation is currently limited to only one virtual router per physical node.

Data-Plane Hypervisor

As explained in Section 4.4, VROOM extends the standard control plane/data plane interface to a migration-aware data-plane hypervisor. Our prototype presents a rudimentary data-plane hypervisor implementation which only supports FIB updates. (A full-fledged data-plane hypervisor would also allow the configuration of other data plane states.) We implemented the `virtd` program as the data-plane hypervisor. `virtd` runs in the VE0 and provides an interface for virtual routers to install/remove routes in the shared data plane, as shown in Figure 4.5. We also implemented the `shadowd` program that runs inside each VE and pushes route updates from the control plane to the FIB through `virtd`.

We run the Quagga routing software suite [79] as the control plane inside each VE. Quagga supports many routing protocols, including BGP and OSPF. In addition to the included protocols, Quagga provides an interface in `zebra`, its routing manager, to allow the addition of new protocol daemons. We made use of this interface to implement `shadowd` as a client of `zebra`. `zebra` provides clients with both the ability to notify `zebra` of route changes and to be notified of route changes. As `shadowd` is not a routing protocol but simply a shadowing daemon, it uses only

the route redistribution capability. Through this interface, `shadowd` is notified of any changes in the RIB and immediately mirrors them to `virtd` using remote procedure calls (RPCs). Each `shadowd` instance is configured with a unique ID (e.g., the ID of the virtual router), which is included in every message it sends to `virtd`. Based on this ID, `virtd` can correctly install/remove routes in the corresponding FIB upon receiving updates from a `shadowd` instance. In the SD prototype, this involves using the Linux *iproute2* utility to set a routing table entry. In the HD prototype, this involves using the device driver to write to registers in the NetFPGA.

Dynamic Interface Binding

With the separation of control and data planes, and the sharing of the same data plane among multiple virtual routers, the data path of each virtual router must be set up properly to ensure that (i) data packets can be forwarded according to the right FIB, and (ii) routing messages can be delivered to the right control plane.

We implemented the `bindd` program that meets these requirements by providing two main functions. The first is to set up the mapping between a virtual router's substrate interfaces and its FIB after the virtual router is instantiated or migrated, to ensure correct packet forwarding. (Note that a virtual router's substrate interface could be either a dedicated physical interface or a tunnel interface that shares the same physical interface with other tunnels.) In the SD prototype, `bindd` establishes this binding by using the routing policy management function (i.e., "ip rule") provided by the Linux *iproute2* utility. As previously mentioned, the HD prototype is currently limited to a single table. Once NetFPGA supports virtualization, a mechanism similar to the "ip rule" function can be used to bind the interfaces with the FIBs.

The second function of `bindd` is to bind the substrate interfaces with the virtual interfaces of the control plane. In both prototypes, this binding is achieved by connecting each pair of substrate and virtual interfaces to a different bridge using the Linux *brctl* utility. In the HD prototype, each of the four physical ports on the NetFPGA is presented to Linux as a separate physical interface, so

packets destined to the control plane of a local VE are passed from the NetFPGA to Linux through the corresponding interface.

4.5.2 Realizing Virtual Router Migration

The above mechanisms set the foundation for VROOM virtual router migration in the OpenVZ environment. We now describe the implementations of data-plane cloning, remote control plane, and double data planes.

Although migration is transparent to the routing processes running in the VE, `shadowd` needs to be notified at the end of the control plane migration in order to start the "data plane cloning". We implemented a function in `shadowd` that, when called, triggers `shadowd` to request `zebra` to resend all the routes and then push them down to `virtd` to repopulate the FIB. Note that `virtd` runs on a fixed (private) IP address and a fixed port on each physical node. Therefore, after a virtual router is migrated to a new physical node, the route updates sent by its `shadowd` can be seamlessly routed to the local `virtd` instance on the new node.

To enable a migrated control plane to continue updating the old FIB (i.e., to act as a "remote control plane"), we implemented in `virtd` the ability to forward route updates to another `virtd` instance using the same RPC mechanism that is used by `shadowd`. As soon as virtual router VR1 is migrated from node A to node B, the migration script notifies the `virtd` instance on B of A's IP address and VR1's ID. B's `virtd`, besides updating the new FIB, starts forwarding the route updates from VR1's control plane to A, whose `virtd` then updates VR1's old FIB. After all of VR1's links are migrated, the old data plane is no longer used, so B's `virtd` is notified to stop forwarding updates. With B's `virtd` updating both the old and new FIBs of VR1 (i.e., the "double data planes"), the two data planes can forward packets during the asynchronous link migration process.

Note that the data-plane hypervisor implementation makes the the control planes unaware of the details of a particular underlying data plane. As as result, migration can occur between any

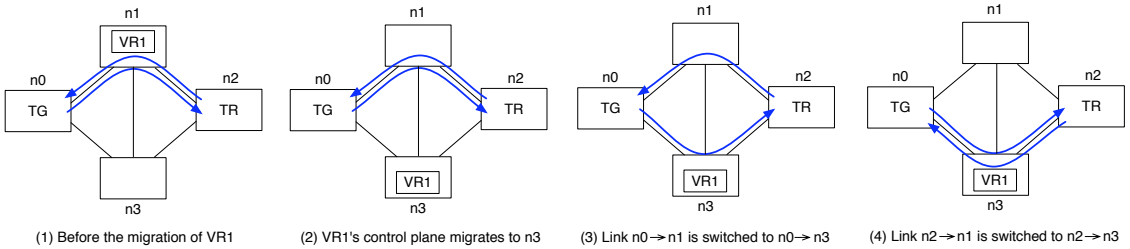


Figure 4.6: The diamond testbed and the experiment process

combination of our HD and SD prototypes (i.e. SD to SD, HD to HD, SD to HD, and HD to SD).

4.6 Evaluation

In this section, we evaluate the performance of VROOM using our SD and HD prototype routers. We first measure the performance of the basic functions of the migration process individually, and then place a VROOM router in a network and evaluate the effect its migration has on the data and control planes. Specifically, we answer the following two questions:

1. *What is the impact of virtual router migration on data forwarding?* Our evaluation shows that it is important to have bandwidth isolation between migration traffic and data traffic. With separate bandwidth, migration based on an HD router has **no** performance impact on data forwarding. Migration based on a SD router introduces minimal delay increase and no packet loss to data traffic.

2. *What is the impact of virtual router migration on routing protocols?* Our evaluation shows that a virtual router running only OSPF in an Abilene-topology network can support 1-second OSPF *hello-interval* without losing protocol adjacencies during migration. The same router loaded with an additional full Internet BGP routing table can support a minimal OSPF *hello-interval* of 2 seconds without losing OSPF or BGP adjacencies.

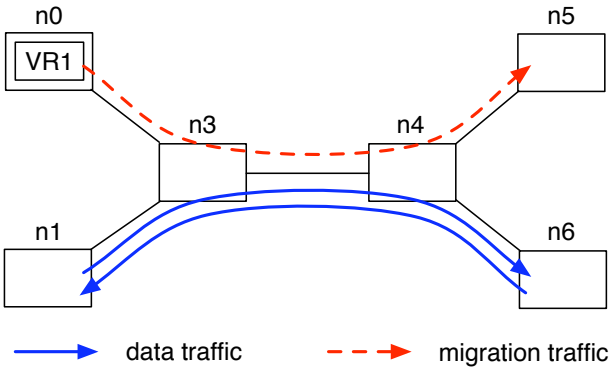


Figure 4.7: The dumbbell testbed for studying bandwidth contention between migration traffic and data traffic. Virtual router VR1 migrates from n0 to n5. Round-trip traffic is sent between n1 and n6.

4.6.1 Methodology

Our evaluation involved experiments conducted in the Emulab tesbed [32]. We primarily used PC3000 machines as the physical nodes in our experiments. The PC3000 is an Intel Xeon 3.0 GHz 64-bit platform with 2GB RAM and five Gigabit Ethernet NICs. For the HD prototype, each physical node was additionally equipped with a NetFPGA card. All nodes in our experiments were running an OpenVZ patched Linux kernel 2.6.18-ovz028stab049.1. For a few experiments we also used the lower performance PC850 physical nodes, built on an Intel Pentium III 850MHz platform with 512MB RAM and five 100Mbps Ethernet NICs.

We used three different testbed topologies in our experiments:

The diamond testbed: We use the 4-node diamond-topology testbed (Figure 4.6) to evaluate the performance of individual migration functions and the impact of migration on the data plane. The testbed has two different configurations, which have the same type of machines as physical node n0 and n2, but differ in the hardware on node n1 and n3. In the *SD* configuration, n1 and n3 are regular PCs on which we install our SD prototype routers. In the *HD* configuration, n1 and n3 are PCs each with a NetFPGA card, on which we install our HD prototype routers. In the experiments, virtual router VR1 is migrated from n1 to n3 through link n1→n3.

The dumbbell testbed: We use a 6-node dumbbell-shaped testbed to study the bandwidth con-

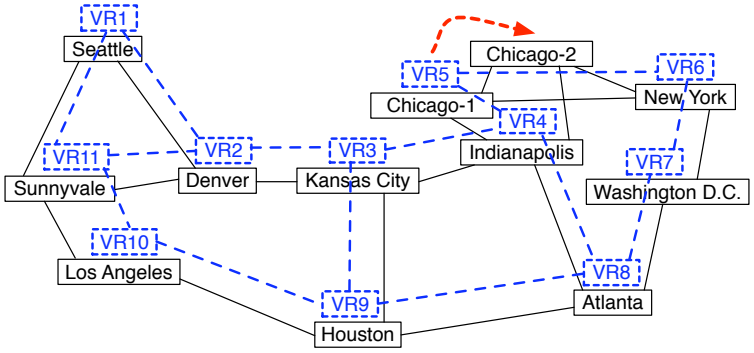


Figure 4.8: The Abilene testbed

Table 4.1: The memory dump file size of virtual router with different numbers of OSPF routes

Routes	0	10k	100k	200k	300k	400k	500k
Size (MB)	3.2	24.2	46.4	58.4	71.1	97.3	124.1

tention between migration traffic and data traffic. In the testbed, round-trip UDP data traffic is sent between a pair of nodes while a virtual router is being migrated between another pair of nodes. The migration traffic and data traffic are forced to share the same physical link, as shown in Figure 4.7.

The Abilene testbed: We use a 12-node testbed (Figure 4.8) to evaluate the impact of migration on the control plane. It has a topology similar to the 11-node Abilene network backbone [2]. The only difference is that we add an additional physical node (Chicago-2), to which the virtual router on Chicago-1 (V5) is migrated. Figure 4.8 shows the initial topology of the virtual network, where 11 virtual routers (V1 to V11) run on the 11 physical nodes (except Chicago-2) respectively.

4.6.2 Performance of Migration Steps

In this subsection, we evaluate the performance of the two main migration functions of the prototypes—memory copy and FIB repopulation.

Memory copy: To evaluate memory copy time relative to the memory usage of the virtual router, we load the `ospfd` in VR1 with different numbers of routes. Table 4.1 lists the respective memory dump file sizes of VR1. Figure 4.9 shows the total time it takes to complete the memory-copy step,

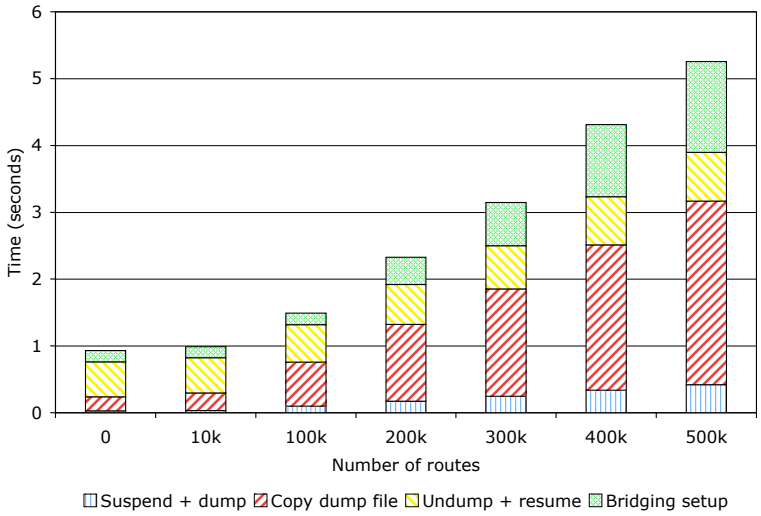


Figure 4.9: Virtual router memory-copy time with different numbers of routes

including (1) suspend/dump VR1 on n1, (2) copy the dump file from n1 to n3, (3) resume VR1 on n3, and (4) set up the bridging (interface binding) for VR1 on n3. We observe that as the number of routes becomes larger, the time it takes to copy the dump file becomes the dominating factor of the total memory copy time. We also note that when the memory usage becomes large, the bridging setup time also grows significantly. This is likely due to CPU contention with the virtual router restoration process, which happens at the same time.

FIB repopulation: We now measure the time it takes VR1 to repopulate the new FIB on n3 after its migration. In this experiment, we configure the virtual router with different numbers of static routes and measure the time it takes to install all the routes into the FIB in the software or hardware data plane. Table 4.2 compares the FIB update time and total time for FIB repopulation. FIB update time is the time `virtd` takes to install route entries into the FIB, while total time also includes the time for `shadowd` to send the routes to `virtd`. Our results show that installing a FIB entry into the NetFPGA hardware (7.4 microseconds) is over 250 times faster than installing a FIB entry into the Linux kernel routing table (1.94 milliseconds). As can be expected the update time increases linearly with the number of routes.

Table 4.2: The FIB repopulating time of the SD and HD prototypes

Data plane type	Software data plane (SD)				Hardware data plane (HD)			
Number of routes	100	1k	10k	15k	100	1k	10k	15k
FIB update time (sec)	0.1946	1.9318	19.3996	31.2113	0.0008	0.0074	0.0738	0.1106
Total time (sec)	0.2110	2.0880	20.9851	33.8988	0.0102	0.0973	0.9634	1.4399

4.6.3 Data Plane Impact

In this subsection, we evaluate the influence router migration has on data traffic. We run our tests in both the HD and SD cases and compare the results. We also study the importance of having bandwidth isolation between the migration and data traffic.

Zero impact: HD router with separate migration bandwidth

We first evaluate the data plane performance impact of migrating a virtual router from our HD prototype router. We configure the HD testbed such that the migration traffic from n1 to n3 goes through the direct link n1→n3, eliminating any potential bandwidth contention between the migration traffic and data traffic.

We run the D-ITG traffic generator [29] on n0 and n2 to generate round-trip UDP traffic. Our evaluation shows that, even with the maximum packet rate the D-ITG traffic generator on n0 can handle (sending and receiving 64-byte UDP packets at 91k packets/s), migrating the virtual router VR1 from n1 to n3 (including the control plane migration and link migration) does not have any performance impact on the data traffic it is forwarding—there is no delay increase or packet loss[4]. These results are not surprising, as the packet forwarding is handled by the NetFPGA, whereas the migration is handled by the CPU. This experiment demonstrates that hardware routers with separate migration bandwidth can migrate virtual routers with zero impact on data traffic.

[4]We hard-wire the MAC addresses of adjacent interfaces on each physical nodes to eliminate the need for ARP request/response during link migration.

Minimal impact: SD router with separate migration bandwidth

In the SD router case, CPU is the resource that could potentially become scarce during migration, because the control plane and data plane of a virtual router share the same CPU. We now study the case in which migration and packet forwarding together saturate the CPU of the physical node. As with the HD experiments above, we use link n1→n3 for the migration traffic to eliminate any bandwidth contention.

In order to create a CPU bottleneck on n1, we use PC3000 machines on n0 and n2 and use lower performance PC850 machines on n1 and n3. We migrate VR1 from n1 to n3 while sending round-trip UDP data traffic between nodes n0 and n2. We vary the packet rate of the data traffic from 1k to 30k packets/s and observe the performance impact the data traffic experiences due to the migration. (30k packets/s is the maximum bi-directional packet rate a PC850 machine can handle without dropping packets.)

Somewhat surprisingly, the delay increase caused by the migration is only noticeable when the packet rate is relatively low. When the UDP packet rate is at 5k packets/s, the control plane migration causes sporadic round-trip delay increases up to 3.7%. However, when the packet rate is higher (e.g., 25k packets/s), the change in delay during the migration is negligible ($< 0.4\%$).

This is because the packet forwarding is handled by kernel threads, whereas the OpenVZ migration is handled by user-level processes (e.g., `ssh`, `rsync`, etc.). Although kernel threads have higher priority than user-level processes in scheduling, Linux has a mechanism that prevents user-level processes from starving when the packet rate is high. This explains the delay increase when migration is in progress. However, the higher the packet rate is, the more frequently the user-level migration processes are interrupted, and more frequently the packet handler is called. Therefore, the higher the packet rate gets, the less additional delay the migration processes add to the packet forwarding. This explains why when the packet rate is 25k packets/s, the delay increase caused by migration becomes negligible. This also explains why migration does not cause any packet drops in the experiments. Finally, our experiments indicate that the link migration does not affect

Table 4.3: Packet loss rate of the data traffic, with and without migration traffic

Data traffic rate (Mbps)	500	600	700	800	900
Baseline (%)	0	0	0	0	0.09
w/ migration traffic (%)	0	0	0.04	0.14	0.29

forwarding delay.

Reserved migration bandwidth is important

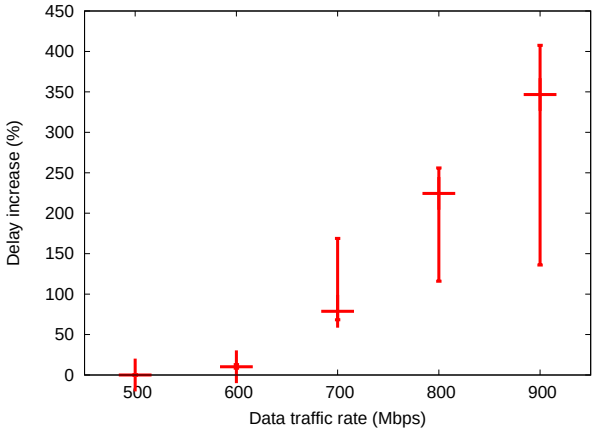


Figure 4.10: Delay increase of the data traffic, due to bandwidth contention with migration traffic

In 4.6.3 and 4.6.3, migration traffic is given its own link (i.e., has separate bandwidth). Here we study the importance of this requirement and the performance implications for data traffic if it is not met.

We use the dumbbell testbed in this experiment, where migration traffic and data traffic share the same bottleneck link. We load the `ospfd` of a virtual router with 250k routes. We start the data traffic rate from 500 Mbps, and gradually increase it to 900 Mbps. Because OpenVZ uses TCP (`scp`) for memory copy, the migration traffic only receives the left-over bandwidth of the UDP data traffic. As the available bandwidth decreases to below 300 Mbps, the migration time increases, which translates into a longer control-plane downtime for the virtual router.

Figure 4.10 compares the delay increase of the data traffic at different rates. Both the aver-

age delay and the delay jitter increase dramatically as the bandwidth contention becomes severe. Table 4.3 compares the packet loss rates of the data traffic at different rates, with and without migration traffic. Not surprisingly, bandwidth contention (i.e., data traffic rate $\geq$ 700 Mbps) causes data packet loss. The above results indicate that in order to minimize the control-plane downtime of the virtual router, and to eliminate the performance impact to data traffic, operators should provide separate bandwidth for the migration traffic.

4.6.4 Control Plane Impact

In this subsection, we investigate the control plane dynamics introduced by router migration, especially how migration affects the protocol adjacencies. We assume a backbone network running MPLS, in which its edge routers run OSPF and BGP, while its core routers run only OSPF. Our results show that, with default timers, protocol adjacencies of both OSPF and BGP are kept intact, and at most one OSPF LSA retransmission is needed in the worst case.

Core Router Migration

We configure virtual routers VR1, VR6, VR8 and VR10 on the Abilene testbed (Figure 4.8) as edge routers, and the remaining virtual routers as core routers. By migrating VR5 from physical node Chicago-1 to Chicago-2, we observe the impact of migrating a core router on OSPF dynamics.

No events during migration: We first look at the case in which there are no network events during the migration. Our experiment results show that the control-plane downtime of VR5 is between 0.924 and 1.008 seconds, with an average of 0.972 seconds over 10 runs.

We start with the default OSPF timers of Cisco routers: *hello-interval* of 10 seconds and *dead-interval* of 40 seconds. We then reduce the *hello-interval* to 5, 2, and 1 second in subsequent runs, while keeping the *dead-interval* equal to four times the *hello-interval*. We find that the OSPF adjacencies between the migrating VR5 and its neighbors (VR4 and VR6) stay up in all cases. Even in the most restrictive 1-second *hello-interval* case, at most one OSPF hello message is lost

and VR5 comes back up on Chicago-2 before its neighbors' dead timers expire.

Events happen during migration: We then investigate the case in which there are events during the migration and the migrating router VR5 misses the LSAs triggered by the events. We trigger new LSAs by flapping the link between VR2 and VR3. We observe that VR5 misses an LSA when the LSA is generated during VR5's 1-second downtime. In such a case, VR5 gets a retransmission of the missing LSA 5 seconds later, which is the default LSA *retransmit-interval*.

We then reduce the LSA *retransmit-interval* from 5 seconds to 1 second, in order to reduce the time that VR5 may have a stale view of the network. This change brings down the maximum interval between the occurrence of a link flap and VR5's reception of the resulting LSA to 2 seconds (i.e., the 1 second control plane downtime plus the 1 second LSA *retransmit-interval*).

Edge Router Migration

Here we configure VR5 as the fifth edge router in the network that runs BGP in addition to OSPF. VR5 receives a full Internet BGP routing table with 255k routes (obtained from RouteViewson Dec 12, 2007) from an eBGP peer that is not included in Figure 4.8, and it forms an iBGP full mesh with the other four edge routers.

With the addition of a full BGP table, the memory dump file size grows from 3.2 MB to 76.0 MB. As a result, it takes longer to suspend/dump the virtual router, copy over its dump file, and resume it. The average downtime of the control plane during migration increases to between 3.484 and 3.594 seconds, with an average of 3.560 seconds over 10 runs. We observe that all of VR5's BGP sessions stay intact during its migration. The minimal integer *hello-interval* VR5 can support without breaking its OSPF adjacencies during migration is 2 seconds (with *dead-interval* set to 8 seconds). In practice, ISPs are unlikely to set the timers much lower than the default values, in order to shield themselves from faulty links or equipment.

4.7 Migration Scheduling

Besides the question of migration mechanisms ("how to migrate"), another important question is the migration scheduling ("where to migrate"). Here we briefly discuss the constraints that need to be considered when scheduling migration and several optimization formulations that can be used in VROOM migration scheduling.

When deciding where to migrate a virtual router, several physical constraints need to be taken into consideration. First of all, an "eligible" destination physical router for migration must use a *software platform* compatible with the original physical router, and have similar (or greater) *capabilities* (such as the number of access control lists supported). In addition, the destination physical router must have sufficient resources available, including *processing power* (whether the physical router is already hosting the maximum number of virtual routers it can support) and *link capacity* (whether the links connected to the physical router have enough unused bandwidth to handle the migrating virtual router's traffic load). Furthermore, the *redundancy* requirement of the virtual router also needs to be considered—today a router is usually connected to two different routers (one as primary and the other as backup) for redundancy. If the primary and backup are migrated to the same node, physical redundancy will be lost.

Fortunately, ISPs typically leave enough "head room" in link capacities to absorb increased traffic volume. Additionally, most ISPs use routers from one or two vendors, with a small number of models, which leaves a large number of eligible physical routers to be chosen for the migration.

Given a physical router that requires maintenance, the question of where to migrate the virtual routers it currently hosts can be formulated as an optimization problem, subject to all the above constraints. Depending on the preference of the operator, different objectives can be used to pick the best destination router, such as minimizing the overall CPU load of the physical router, minimizing the maximum load of physical links in the network, minimizing the stretch (i.e., latency increase) of virtual links introduced by the migration, or maximizing the reliability of the network

(e.g., the ability to survive the failure of any physical node or link). However, finding optimal solutions to these problems may be computationally intractable. Fortunately, simple local-search algorithms should perform reasonably well, since the number of physical routers to consider is limited (e.g., to hundreds or small thousands, even for large ISPs) and finding a "good" solution (rather than an optimal one) is acceptable in practice. In the case of power savings, we take the power prices in different geographic locations into account and try to minimize power consumption with a certain migration granularity (e.g., once every hour, according to the hourly traffic matrices).

4.8 Related Work

VROOM's motivation is similar, in part, to that of the RouterFarm work [4], namely, to reduce the impact of planned maintenance by migrating router functionality from one place in the network to another. However, RouterFarm essentially performs a "cold restart", compared to VROOM's live ("hot") migration. Specifically, in RouterFarm router migration is realized by re-instantiating a router instance at the new location, which not only requires router reconfiguration, but also introduces inevitable downtime in both the control and data planes. In VROOM, on the other hand, we perform *live* router migration without reconfiguration or discernible disruption. In our earlier prototype of VROOM [106], router migration was realized by directly using the standard virtual machine migration capability provided by Xen [9], which lacked the control and data plane separation presented in this chapter. As a result, it involved data-plane downtime during the migration process.

Recent advances in virtual machine technologies and their live migration capabilities [27, 73] have been leveraged in server-management tools, primarily in data centers. For example, Sandpiper [109] automatically migrates virtual servers across a pool of physical servers to alleviate hotspots. Usher [69] allows administrators to express a variety of policies for managing clusters of virtual servers. Remus [28] uses asynchronous virtual machine replication to provide high avail-

ability to server in the face of hardware failures. In contrast, VROOM focuses on leveraging live migration techniques to simplify management in the networking domain.

Network virtualization has been proposed in various contexts. Early work includes the "switchlets" concept, in which ATM switches are partitioned to enable dynamic creation of virtual networks [95]. More recently, the CABO architecture proposes to use virtualization as a means to enable multiple service providers to share the same physical infrastructure [34]. Outside the research community, router virtualization has already become available in several forms in commercial routers [25, 57]. In VROOM, we take an additional step not only to virtualize the router functionality, but also to decouple the virtualized router from its physical host and enable it to migrate.

VROOM also relates to recent work on minimizing transient routing disruptions during planned maintenance. A measurement study of a large ISP showed that more than half of routing changes were planned in advance [55]. Network operators can limit the disruption by reconfiguring the routing protocols to direct traffic away from the equipment undergoing maintenance [93, 41]. In addition, extensions to the routing protocols can allow a router to continue forwarding packets in the data plane while reinstalling or rebooting the control-plane software [90, 21]. However, these techniques require changes to the logical configuration or the routing software, respectively. In contrast, VROOM hides the effects of physical topology changes in the first place, obviating the need for point solutions that increase system complexity while enabling new network-management capabilities, as discussed in the next section.

4.9 Summary

VROOM is a new network-management primitive that supports live migration of virtual routers from one physical router to another. To minimize disruptions, VROOM allows the migrated control plane to clone the data-plane state at the new location while continuing to update the state at

the old location. VROOM temporarily forwards packets using both data planes to support asynchronous migration of the links. These designs are readily applicable to commercial router platforms. Experiments with our prototype system demonstrate that VROOM does not disrupt the data plane and only briefly freezes the control plane. In the unlikely scenario that a control-plane event occurs during the freeze, the effects are largely hidden by existing mechanisms for retransmitting routing-protocol messages. VROOM provides a new, disruption-free mechanism to handle a broad range of network management tasks, such as planned maintenance, service deployment and power savings.

Chapter 5

Conclusion

Today's ISPs face a number of major challenges in better managing routing in their networks to provide competitive services, including (1) maintaining the stability of the *global* Internet while meeting the increasing demands from its customers for diverse routes, (2) supporting intuitive and more flexible routing policy configuration in bilateral contractual relationships with its *neighbors*, and (3) making network maintenance and other network management operations in their *own* networks easier and less disruptive to routing protocols and data traffic. This dissertation takes a *principled* approach to addressing these challenges with a set of new abstractions, as well as theoretical results and systems guided by these abstractions. In this chapter, we first summarize the contributions of this dissertation in Section 5.1. We then discuss the synergy between Morpheus and VROOM and the additional benefits an ISP can get by deploying the two systems together in Section 5.2. We briefly propose some future research directions based on the work done in this dissertation and conclude in Section 5.3.

5.1 Summary of Contributions

This dissertation identified three major routing management challenges faced by ISPs today and presented corresponding theoretical and/or system solutions.

First, we observed that there are many useful routing policies that cannot be realized (configured) today, and argued that it is mainly caused by the "one-route-fits-all" BGP route-selection model. Acknowledging the fact that different customers of an ISP are increasingly likely to prefer routes with different properties, we proposed the abstraction of a "neighbor-specific route selection problem" and a corresponding "Neighbor-Specific BGP" (NS-BGP) model that enables an ISP to offer customized route-selection services to different neighbors. We also proved a surprisingly positive result that, comparing to conventional BGP, a less restrictive sufficient condition can guarantee the stability of the more flexible NS-BGP. Our stability conditions allow an AS to select *any* exportable routes for its neighbors without compromising global stability. This result provides a new understanding of the fundamental trade-off between local policy flexibility and global routing stability. We also show that NS-BGP remains stable even in partial deployment and in the presence of network failures, as long as the stability conditions are followed. As as result, any individual ISP can deploy NS-BGP independently by modifying how routes are selected and disseminated *within* its network, without modifying the BGP message format or requiring collaboration from neighboring domains.

Second, we argued that the notorious difficulty of routing policy configuration is mainly due to two reasons: the mismatch between how the policies are *specified* and how they are *implemented*; as well as the counterintuitive and restrictive BGP configuration interface. We presented the design, implementation and evaluation of Morpheus, a routing control platform that supports customized route selection (NS-BGP) and simplifies policy configuration through an expressive yet intuitive configuration interface. Morpheus enables network operators to easily define new policy objectives, make flexible trade-offs between objectives, and therefore realize a much broader range of

routing policies than conventional BGP.

Finally, we argued that the traffic disruption caused by routine planned maintenance and other network management tasks is due to the *unnecessary* routing configuration changes used as a tool to help physical network changes. Hence, we propose the abstraction of the separation between "physical" and "logical" configurations of routers, which leads us to the design and prototype implementation of "virtual router migration" (VROOM), a new, generic technique to simplify and enable a broad range of network management tasks, from planned maintenance to reducing power consumption.

Collectively, the contributions of the dissertation provide simple system solutions for an ISP to autonomously manage its routing more flexibly and effectively without affecting global routing stability.

5.2 The Synergy of Deploying Morpheus and VROOM Together

Depending on its need, an ISP can choose to deploy only Morpheus or VROOM. However, the two systems together bring additional synergies to each other if deployed together.

On the one hand, Morpheus makes the virtual router migration process of VROOM faster. This is because with Morpheus, individual routers no longer hold BGP states in their control plane (as these states are maintained in the Morpheus servers instead); therefore, only IGP states need to be copied during migration, which has far smaller memory footprint compared to BGP states and can be transferred very quickly. On the other hand, with VROOM, an ISP no longer needs to make configuration changes to Morpheus (and other routers) to assist physical network changes. VROOM can also help reduce the churn of routing updates in a Morpheus-enabled ISP as many network management tasks such as planned maintenance no longer cause protocol reconvergence.

When deployed together, Morpheus and VROOM simplify the management of an ISP by separating different concerns — Morpheus enables network operators to focus on realizing desir-

able routing policies without thinking about underlying physical changes to the network, whereas VROOM allows network operators to carry out physical changes to the network without worrying about disrupting routing protocol adjacencies and data traffic. As a result, each individual task becomes less complicated, less error-prone and more predictable.

5.3 Open Issues and Future Work

The work presented in this dissertation raised a number of questions that deserve further investigation in the future.

5.3.1 Using Morpheus and VROOM to Handle Traffic Engineering

Most policy objectives can be expressed in terms of ratings (preferences) for individual routes. A notable exception is traffic engineering (TE), since the total traffic on each link in the network depends on the mixture of traffic to many different destinations. Today, network operators perform TE by tuning the IGP link weights or configuring MPLS tunnels (in the case of MPLS TE) to move traffic away from congested links. With Morpheus, the network operators can also configure the egress-point rankings to manipulate the flow of traffic. In addition, although some customers will subscribe to customized routes, the remaining customers will still use whatever paths the ISP selects as the "default". Controlling the route-selection process for the default customers gives the ISP substantial leeway to perform TE. As such, providing greater flexibility in path selection serves as an enabler for effective traffic engineering. We believe that exploring these issues in greater depth is a promising avenue for future research.

VROOM provides another interesting alternative to the current TE practice—using virtual router migration as a TE primitive. When a physical link connected to a physical router is overloaded, a portion of the traffic can be offloaded by migrating one or more virtual routers running on the physical router (and the corresponding virtual links) to a different node. This mechanism

could be more appealing than other mechanisms that involve protocol reconfiguration and reconvergence, especially for alleviating "flash crowd" type of traffic surge that lasts for a short period of time (so that traffic taking a slight detour from the shortest paths computed by IGP is less of a concern.)

5.3.2 The Evolving Functions of Routers

Our research on VROOM raises several broader questions about the design of future routers and the relationship with the underlying transport network. Recent innovations in transport networks support rapid set-up and tear-down of links, enabling the network topology to change underneath the IP routers. Dynamic topologies coupled with VROOM's migration of the control plane and cloning of the data plane make the router an increasingly ephemeral concept, not tied to a particular location or piece of hardware. Future work on router hypervisors could take this idea one step further. Just as today's commercial routers have a clear separation between the control and data planes, future routers could decouple the control-plane software (executables) and the control-plane states (e.g., RIBs and state in the state machine for each routing-protocol adjacency, etc.). Such a "control-plane hypervisor" would make it easier to upgrade router software while retaining the states, and for virtual routers to migrate between different kinds of physical router platforms (e.g., those from different vendors and / or running different code bases).

5.3.3 Dynamics of NS-BGP

In this dissertation, we focused primarily on the stability conditions of NS-BGP. Another interesting aspect of this new route-selection model is its dynamics, especially the speed of its convergence (compared to conventional BGP). We observe that, at the very minimum, existing proposals (e.g., [19]) that aim at addressing the path exploration problem—the main cause of the slow BGP convergence—can also be applied to NS-BGP. Other issues that have been studied for conventional

BGP, such as complexity results and root-cause analysis, could also be revisited in the context of NS-BGP.

5.4 Concluding Remarks

This dissertation has (1) presented a new, neighbor-specific BGP (NS-BGP) route-selection model that allows an ISP to provide customized routes to its customers, and proved the conditions that guarantee global stability of the Internet when ISPs switch to the NS-BGP model; (3) designed and implemented a routing control platform that supports NS-BGP and much broader range of routing policies (compared to conventional BGP) via a new, intuitive configuration interface; (4) developed a new technique (virtual router migration) that enables many network management tasks to be conducted without disrupting routing protocol adjacencies or data traffic.

At a high-level, the work presented in this dissertation is motivated by revisiting and challenging previous unquestionable assumptions (such as "one-route-fits-all works well enough", "computing BGP routes in a centralized fashion does not scale", "a router is a piece of monolithic equipment"), and is enabled by leveraging theory from other fields (such as the Analytic Hierarchy Process from decision theory) and the advancement of related technologies (such as the steady increase of computational power under Moore's Law, server virtualization, programmable transport network). We believe that, as enabling technologies keep emerging and/or evolving, revisiting assumptions that previous design decisions were based upon and leveraging emerging technologies will remain an effective approach to addressing network management challenges.

Bibliography

[1] `http://www.networksorcery.com/enp/protocol/bgp.htm`.

[2] The Internet2 Network. `http://www.internet2.edu/`.

[3] T. Afferton, R. Doverspike, C. Kalmanek, and K. K. Ramakrishnan. Packet-aware transport for metro networks. *IEEE Communication Magazine*, March 2004.

[4] M. Agrawal, S. Bailey, A. Greenberg, J. Pastor, P. Sebos, S. Seshan, J. van der Merwe, and J. Yates. RouterFarm: Towards a dynamic, manageable network edge. In *Proc. ACM SIGCOMM Workshop on Internet Network Management (INM)*, September 2006.

[5] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra. Routing policy specification language (RPSL). RFC 2622, June 1999.

[6] H. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck, and J. van der Merwe. Anycast CDNs revisited. In *Proc. International World Wide Web Conference*, 2008.

[7] I. Avramopoulos, J. Rexford, and R. Schapire. From optimization to regret minimization and back again. In *Proc. Third Workshop on Tackling Computer System Problems with Machine Learning Techniques (SysML08)*, December 2008.

[8] D. O. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of Internet traffic engineering. RFC 3272, May 2002.

[9] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebar, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proc. Symposium on Operating Systems Principles (SOSP)*, October 2003.

[10] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. Wilfong. Route oscillations in I-BGP with

route reflection. In *Proc. ACM SIGCOMM*, August 2002.

[11] V. Belton. *Multiple Criteria Decision Analysis*. Springer, 2003.

[12] H. Boehm, A. Feldmann, O. Maennel, C. Reiser, and R. Volk. Network-wide inter-domain routing policies: Design and realization. *Draft*, April 2005.

[13] O. Bonaventure, C. Filsfils, and P. Francois. Achieving sub-50 milliseconds recovery upon BGP peering link failures. *IEEE/ACM Trans. Networking*, October 2007.

[14] S. Bryant and P. Pate. Pseudo wire emulation edge-to-edge (PWE3) architecture. RFC 3985, March 2005.

[15] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe. Design and implementation of a Routing Control Platform. In *Proc. Networked Systems Design and Implementation (NSDI)*, May 2005.

[16] M. Caesar and J. Rexford. BGP policies in ISP networks. *IEEE Network Magazine*, October 2005.

[17] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power awareness in network design and routing. In *Proc. IEEE INFOCOM*, 2008.

[18] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *To appear in ACM Computing Surveys*, September 2009.

[19] J. Chandrashekar, Z. Duan, J. Krasky, and Z.-L. Zhang. Limiting path exploration in BGP. In *Proc. IEEE INFOCOM*, March 2005.

[20] C.-K. Chau. Policy-based routing with non-strict preferences. In *Proc. ACM SIGCOMM*, 2006.

[21] E. Chen, R. Fernando, J. Scudder, and Y. Rekhter. Graceful Restart Mechanism for BGP. RFC 4724, January 2007.

[22] Ciena CoreDirector Multiservice Switch. `www.ciena.com`.

[23] Cisco Optimized Edge Routing. `http://www.cisco.com/en/US/products/ps6628/products_ios_protocol_option_home.html`.

[24] MPLS VPN Carrier Supporting Carrier. `http://www.cisco.com/en/US/docs/ios/12_0st/12_0st14/feature/guide/csc.html`.

[25] Cisco Logical Routers. `http://www.cisco.com/en/US/docs/ios_xr_sw/iosxr_r3.2/interfaces/command/reference/hr32lr.html`.

[26] MPLS Fundamentals: Forwarding Labeled Packets. http://www.ciscopress.com/articles/article.asp?p=680824&seqNum=2.

[27] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proc. Networked Systems Design and Implementation (NSDI)*, May 2005.

[28] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield. Remus: High availability via asynchronous virtual machine replication. In *Proc. Networked Systems Design and Implementation (NSDI)*, April 2008.

[29] D-ITG. http://www.grid.unina.it/software/ITG/.

[30] N. Duffield, K. Gopalan, M. R. Hines, A. Shaikh, and J. van der Merwe. Measurement informed route selection. In *Proc. Passive and Active Measurement Conference (Extended Abstract)*, 2007.

[31] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proc. IEEE INFOCOM*, 2001.

[32] Emulab. http://www.emulab.net.

[33] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe. The case for separating routing from routers. In *Proc. ACM SIGCOMM Workshop on Future Direction in Network Architecture*, August 2004.

[34] N. Feamster, L. Gao, and J. Rexford. How to lease the Internet in your spare time. *ACM Computer Communication Review*, pages 61–64, January 2007.

[35] N. Feamster, R. Johari, and H. Balakrishnan. Implications of autonomy for the expressiveness of policy routing. In *Proc. ACM SIGCOMM*, August 2005.

[36] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. *Distributed Computing*, 18:61–72, 2005.

[37] J. Feigenbaum, R. Sami, and S. Shenker. Mechanism design for policy routing. *Distributed Computing*, 18(4):293–305, 2006.

[38] A. Feldmann and J. Rexford. IP network configuration for intradomain traffic engineering. *IEEE Network Magazine*, pages 46–57, September/October 2001.

[39] S. Fischer, N. Kammenhuber, and A. Feldmann. REPLEX — dynamic traffic engineering based on

wardrop routing policies. In *Proc. CoNext*, December 2006.

[40] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proc. IEEE INFOCOM*, March 2000.

[41] P. Francois, M. Shand, and O. Bonaventure. Disruption-free topology reconfiguration in OSPF networks. In *Proc. IEEE INFOCOM*, May 2007.

[42] L. Gao, T. G. Griffin, and J. Rexford. Inherently safe backup routing with BGP. In *Proc. IEEE INFOCOM*, pages 547–556, April 2001.

[43] L. Gao and J. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Trans. Networking*, December 2001.

[44] S. Goldberg, S. Halevi, A. Jaggard, V. Ramachandran, and R. Wright. Rationality and traffic attraction: Incentives for honestly announcing paths in BGP. In *Proc. ACM SIGCOMM*, August 2008.

[45] T. Griffin and G. Huston. BGP wedgies. RFC 4264, November 2005.

[46] T. Griffin, A. Jaggard, and V. Ramachandran. Design principles of policy languages for path vector protocols. In *Proc. ACM SIGCOMM*, pages 61–72, Karlsruhe, Germany, August 2003.

[47] T. Griffin, F. B. Shepherd, and G. Wilfong. Policy disputes in path-vector protocols. In *Proc. International Conference on Network Protocols (ICNP)*, November 1999.

[48] T. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Networking*, 10(1):232–243, 2002.

[49] T. Griffin and G. Wilfong. An analysis of BGP convergence properties. In *Proc. ACM SIGCOMM*, September 1999.

[50] T. Griffin and G. Wilfong. Analysis of the MED oscillation problem in BGP. In *Proc. International Conference on Network Protocols (ICNP)*, Paris, France, November 2002.

[51] T. Griffin and G. Wilfong. On the correctness of IBGP configuration. *SIGCOMM Comput. Commun. Rev.*, 32(4):17–29, 2002.

[52] M. Gupta and S. Singh. Greening of the Internet. In *Proc. ACM SIGCOMM*, August 2003.

[53] M. Handley, E. Kohler, A. Ghosh, O. Hodson, and P. Radoslavov. Designing extensible IP router software. In *Proc. Networked Systems Design and Implementation (NSDI)*, May 2005.

[54] J. He, M. Suchara, M. Bresler, J. Rexford, and M. Chiang. Rethinking Internet traffic management:

From multiple decompositions to a practical protocol. In *Proc. CoNext*, December 2007.

[55] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot. Feasibility of IP restoration in a tier-1 backbone. *IEEE Network Magazine*, Mar 2004.

[56] IETF. *Advertisement of Multiple Paths in BGP*, July 2008. `http://tools.ietf.org/html/draft-walton-bgp-add-paths-06`. Internet Draft. Expires January 2009.

[57] Juniper Logical Routers. `http://www.juniper.net/techpubs/software/junos/junos85/feature-guide-85/id-11139212.html`.

[58] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proc. ACM SIGCOMM*, August 2005.

[59] J. Karlin, S. Forrest, and J. Rexford. Autonomous security for autonomous systems. *Computer Networks*, October 2008.

[60] Z. Kerravala. Configuration management delivers business resiliency. The Yankee Group, November 2002.

[61] R. R. Kompella, S. Singh, and G. Varghese. On scalable attack detection in the network. In *Proc. Internet Measurement Conference (IMC)*, October 2004.

[62] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. *IEEE/ACM Trans. Networking*, 9(3):293–306, June 2001.

[63] C. Labovitz, R. Malan, and F. Jahanian. Internet routing instability. *IEEE/ACM Trans. Networking*, 6(5):515–528, October 1998.

[64] C. Labovitz, R. Malan, and F. Jahanian. Origins of pathological Internet routing instability. In *Proc. IEEE INFOCOM*, March 1999.

[65] C. Labovitz, R. Wattenhofer, S. Venkatachary, and A. Ahuja. The impact of Internet policy and topology on delayed routing convergence. In *Proc. IEEE INFOCOM*, April 2001.

[66] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proc. ACM SIGCOMM*, August 2004.

[67] H. Levin, M. Schapira, and A. Zohar. Interdomain routing and games. In *Proc. ACM Symposium on Theory of Computing (STOC)*, 2008.

[68] R. Mahajan, D. Wetherall, and T. Anderson. Mutually controlled routing with independent ISPs. In

Proc. Networked Systems Design and Implementation (NSDI), 2007.

[69] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker. Usher: An extensible framework for managing clusters of virtual machines. In *Proc. USENIX LISA Conference*, November 2007.

[70] W. Muhlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig. Building an AS-topology model that captures route diversity. In *Proc. ACM SIGCOMM*, 2006.

[71] Number of BGP routes a large ISP sees in total. Discussion on NANOG mailing list, `http://www.merit.edu/mail.archives/nanog/2007-04/msg00502.html`.

[72] NetFPGA. `http://yuba.stanford.edu/NetFPGA/`.

[73] OpenVZ. `http://openvz.org`.

[74] `http://www.lightreading.com/document.asp?doc_id=22223`.

[75] `http://www.researchandmarkets.com/reports/448691/demystifying_opex_and_capex_budgets_feedback`.

[76] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT press, 1994.

[77] I. Pepelnjak and J. Guichard. *MPLS and VPN Architectures*. Cisco Press, 2000.

[78] Average retail price of electricity. `http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_a.html`.

[79] Quagga Routing Suite. `http://www.quagga.net`.

[80] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (BGP-4). RFC 4271, January 2006.

[81] J. Rexford, G. Wilfong, and R. Zhang-Shen. Routers of the AS, unite! Guaranteeing a network realizes its routing policy. Technical Report TR-847-09, Dept. of Computer Science, Princeton Univ., February 2009.

[82] A. Rostami and E. Sargent. An optical integrated system for implementation of NxM optical cross-connect, beam splitter, mux/demux and combiner. International Journal of Computer Science and Network Security (IJCSNS), July 2006.

[83] K. Roth, F. Goldstein, and J. Kleinman. Energy consumption by office and telecommunications equipment in commercial buildings volume I: Energy consumption baseline. National Technical Information Service (NTIS), U.S. Department of Commerce, Springfield, VA 22161, NTIS Number: PB2002-101438, 2002.

[84] The routeviews project. `www.routeviews.org`.

[85] T. L. Saaty. *The Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process, Vol. VI, AHP Series*. RWS Publications, 2000.

[86] R. Sami, M. Schapira, and A. Zohar. Security and selfishness in interdomain routing. Technical report, Leibniz Center for Research in Computer Science, 2008.

[87] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *Proc. ACM SIGCOMM*, 1999.

[88] J. Scudder, R. Fernando, and S. Stuart. BGP monitoring protocol. Internet Draft draft-ietf-grow-bmp-00, November 2008.

[89] V. Sekar, N. Duffield, O. Spatscheck, J. van der Merwe, and H. Zhang. LADS: Large-scale automated DDoS detection system. In *Proc. USENIX*, June 2006.

[90] A. Shaikh, R. Dube, and A. Varma. Avoiding instability during graceful shutdown of multiple OSPF routers. *IEEE/ACM Trans. Networking*, 14(3):532–542, June 2006.

[91] N. Spring, R. Mahajan, and T. Anderson. Quantifying the causes of path inflation. In *Proc. ACM SIGCOMM*, 2003.

[92] H. Tangmunarunkit, R. Govindan, and S. Shenker. Internet path inflation due to policy routing. In *Proc. SPIE ITCom*, 2001.

[93] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in IP networks. In *Proc. ACM SIGMETRICS*, June 2004.

[94] S. Uhlig and S. Tandel. Quantifying the impact of route-reflection on BGP routes diversity inside a tier-1 network. In *Proc. IFIP NETWORKING*, 2006.

[95] J. van der Merwe and I. Leslie. Switchlets and dynamic virtual ATM networks. In *Proc. IFIP/IEEE International Symposium on Integrated Network Management*, May 1997.

[96] J. van der Merwe, et al. Dynamic connectivity management with an intelligent route service control point. In *Proc. ACM SIGCOMM Workshop on Internet Network Management (INM)*, September 2006.

[97] P. Verkaik, D. Pei, T. Scholl, A. Shaikh, A. Snoeren, and J. van der Merwe. Wresting control from BGP: Scalable fine-grained route control. In *Proc. USENIX*, 2007.

[98] C. Villamizar, R. Chandra, and R. Govindan. BGP Route Flap Damping. RFC 2439, November 1998.

[99] VINI. `http://www.vini-veritas.net/`.

[100] D. Walton, A. Retana, E. Chen, and J. Scudder. Advertisement of multiple paths in BGP. Internet Draft draft-ietf-idr-add-paths-00.txt, December 2008.

[101] H. Wang, H. Xie, L. qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. COPE: Traffic engineering in dynamic networks. In *Proc. ACM SIGCOMM*, September 2006.

[102] Y. Wang, I. Avramopoulos, and J. Rexford. Morpheus: Enabling flexible interdomain routing policies. Technical Report TR-802-07, Princeton University, October 2007.

[103] Y. Wang, I. Avramopoulos, and J. Rexford. Design for configurability: Rethinking interdomain routing policies from the ground up. *IEEE Journal on Selected Areas in Communications*, April 2009.

[104] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford. Virtual routers on the move: Live router migration as a network-management primitive. In *Proc. ACM SIGCOMM*, August 2008.

[105] Y. Wang, M. Schapira, and J. Rexford. Neighbor-specific BGP: More flexible routing policies while improving global stability. In *Proc. ACM SIGMETRICS*, June 2009.

[106] Y. Wang, J. van der Merwe, and J. Rexford. VROOM: Virtual ROuters On the Move. In *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking (HotNets)*, Nov 2007.

[107] J. Wei, K. Ramakrishnan, R. Doverspike, and J. Pastor. Convergence through packet-aware transport. *Journal of Optical Networking*, 5(4), April 2006.

[108] D. Wendlandt, I. Avramopoulos, D. Andersen, and J. Rexford. Don't secure routing protocols, secure data delivery. In *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking (HotNets)*, November 2006.

[109] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif. Black-box and Gray-box Strategies for Virtual Machine Migration. In *Proc. Networked Systems Design and Implementation (NSDI)*, April 2007.

[110] XORP: Open Source IP Router. `http://www.xorp.org`.

[111] Y. R. Yang, H. Xie, H. Wang, A. Silberschatz, Y. Liu, L. E. Li, and A. Krishnamurthy. On route selection for interdomain traffic engineering. *IEEE Network Magazine*, November 2005.

[112] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund. Online identification of hierarchical heavy

hitters: Algorithms, evaluation, and applications. In *Proc. Internet Measurement Conference (IMC)*, 2004.